PhpStorm 2016.2 Help

Meet PhpStorm

Welcome to PhpStorm help!

PhpStorm is an Integrated Development Environment for the PHP developers, built on top of IntelliJ IDEA platform. PhpStorm inherits all the web-related functionality of IntelliJ IDEA for editing PHP, HTML, CSS, JavaScript, XML, working with VCS, SQL, plus adds advanced support for the other tools specific to web development.

PhpStorm brings you the following advanced features:

- Intelligent Editor: for PHP, HTML, CSS, JavaScript, and XML, which includes syntax highlighting, documentation lookup, and refactorings.
- Error-Free Coding: on-the-fly code analysis, error highlighting and quick fixes.
- Project and Code Navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.
- VCS Integrations: out-of-the-box support for Subversion, Perforce, Git, and CVS with changelists and merge.
- FTP Sync: update server using FTP or SFTP.
- SQL support: coding assistance, SQL console and Database browser.

JetBrains PhpStorm is cross-platform. It works on Windows, OS X and Linux and brings the whole range of precise developers tools, all tied together to create the convenient development environment .

- If you are completely new to PhpStorm, read Quick Start Guide . The links in this section will lead you to more detailed instructions.
- Before starting with PhpStorm, take a look at the essentials , since they are useful for more productive usage of PhpStorm.
- If you want to learn about the PhpStorm UI, then take a guided tour . The parts Tool Windows and PhpStorm Editor tell you everything you need to know to get a grip of these UI elements.
- If you click Help button on a dialog, you will see the reference page that gives descriptions of controls and fields of each dialog.
- Besides the dialog descriptions, the part Reference also keeps miscellaneous important information, like icons reference , index of menu items , version control reference , etc.
- The tutorials for the newbies are in the part First Steps . They help perform the most important basic tasks with PhpStorm - create and run an application.
- For those who want to learn more about PhpStorm, the part General Guidelines provides information about creating projects, configuring their structure, output etc.
- The more advanced users will find it useful to read the various sections of the part Language and Framework - Specific Guidelines .
- The part Tutorials contains the detailed step-by-step tutorials, which will lead you through specific tasks.
- Finally, the part Getting Help tells about using help topics in the online and built-in variants, applying to support service, reporting issues and sharing your feedback.

System Requirements

On this page:

- Hardware requirements
- System requirements
  - Windows
  - OS X
  - Linux

Tip To get acquainted with PhpStorm, you can download and install its trial version for free. This trial version is available for 30 days, whereupon you need to obtain and register a license.

Hardware requirements

- 1 GB RAM minimum, 2 GB RAM recommended
- 1024x768 minimum screen resolution

System requirements

JRE 1.8 is bundled with distributions for all platforms. You don't need any Java to be installed on your machine in order to run PhpStorm.

Windows

System Requirements	Installation
- Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)	1. Download PhpStorm from the Download page. 2. Run the `PhpStorm-*.exe` file that starts the Installation Wizard. 3. Follow all steps suggested by the wizard. Pay special attention to the corresponding installation options.

OS X

System Requirements	Installation
- OS X 10.5 or higher, including 10.10 - Only 64-bit OS X is supported.	1. Download the `PhpStorm-*.dmg` OS X Disk Image file from the Download page. 2. Double-click the downloaded `PhpStorm-*.dmg` OS X Disk Image file to mount it. 3. Copy PhpStorm to your Applications folder.

Linux

System Requirements	Installation
- OS Linux 64 bit - KDE, GNOME or Unity DE desktop	1. Download the `PhpStorm-*.tar.gz` file from the Download page. 2. Unpack the `PhpStorm-*.tar.gz` file to an empty directory using the following command:

```
tar xfz PhpStorm-*.tar.gz
```

3. Because running PhpStorm from wherever you downloaded the file to may be inconvenient, it is recommended that you move the extracted or unpacked archive folder to the desired location using the `mv` command in one of the following formats:
   -
     ```
     mv <path to extracted
                         archive
                         folder> <new archive
                         folder>
     ```

     For example:

     ```
     mv /downloads/PhpStorm-* my/desired/location
     ```
   -
     ```
     mv <path to PhpStorm-*.tar.gz>
                         <new archive folder>
     ```

     For example:

     ```
     mv /downloads/PhpStorm-*.tar.gz my/desired/location
     ```
4. Switch to the `bin` directory in the new location:

   ```
   cd <new archive folder>/PhpStorm-*/bin
   ```

   For example,

   ```
   cd my/desired/location/PhpStorm-*/bin
   ```
5. Run `phpstorm.sh` from the `bin` subdirectory.

Installing and Launching

On this page:

- Downloading and installing PhpStorm
- Starting PhpStorm on the various platforms
  - Starting PhpStorm on Windows
  - Starting PhpStorm on OS X
  - Starting PhpStorm on Linux
    - Additional launchers
      - Creating a desktop shortcut
- Importing PhpStorm usage preferences from a previous version

Tip To get acquainted with PhpStorm, you can download and install its trial version for free. This trial version is available for 30 days, whereupon you need to obtain and register a license.

Downloading and installing PhpStorm

You can download the latest version of PhpStorm from the JetBrains website . (There are the versions for Windows, OS X and Linux.)

Depending on your operating system:

- Windows: Run the `.exe` file and follow the instructions of PhpStorm Setup wizard.
- OS X: Open the `.dmg` package, and drag PhpStorm to the Applications folder.
- Linux: Unpack the `.tar.gz` archive into any directory within your home directory.

Refer to the section Starting PhpStorm on the various platforms for details.

Starting PhpStorm on the various platforms

Since 2016.1 PhpStorm uses Java 1.8.

Starting PhpStorm on Windows

If during the installation you selected to create a shortcut on your desktop, double-click that shortcut.

Otherwise, go to the `<PhpStorm>\bin` folder (e.g. `C:\Program Files (x86)\JetBrains\IntelliJ IDEA 2016.1\bin`) and run `idea.exe` or `idea.bat` .

See also, Importing PhpStorm usage preferences from a previous version.

Starting PhpStorm on OS X

See Importing PhpStorm usage preferences from a previous version.

Also, if you wish to change JDK PhpStorm is running on, use the action Switch boot JDK . To invoke this action, use Searching Everywhere or Find Action .

Starting PhpStorm on Linux

To launch PhpStorm on Linux, follow these steps:

1. Download the `PhpStorm-*.tar.gz` file from the Download page.
2. Unpack the `PhpStorm-*.tar.gz` file to an empty directory using the following command:

```
tar xfz PhpStorm-*.tar.gz
```

3. Because running PhpStorm from wherever you downloaded the file to may be inconvenient, it is recommended that you move the extracted or unpacked archive folder to the desired location using the `mv` command in one of the following formats:
   -
   ```
   mv <path to extracted
                       archive
                       folder> <new archive
                       folder>
   ```

     For example:

   ```
   mv /downloads/PhpStorm-* my/desired/location
   ```

   -
   ```
   mv <path to PhpStorm-*.tar.gz>
                       <new archive folder>
   ```

     For example:

   ```
   mv /downloads/PhpStorm-*.tar.gz my/desired/location
   ```

4. Switch to the `bin` directory in the new location:

```
cd <new archive folder>/PhpStorm-*/bin
```

For example,

```
cd my/desired/location/PhpStorm-*/bin
```

5. Run `phpstorm.sh` from the `bin` subdirectory.

Additional launchers

Additionally, you can create a desktop shortcut and use it to start PhpStorm:

- Creating a desktop shortcut

See also, Importing PhpStorm usage preferences from a previous version.

Creating a desktop shortcut

To create a desktop shortcut for PhpStorm, use the Create Desktop Entry command. This command is available:

- In the Customize PhpStorm wizard - when you run PhpStorm for the first time.
- On the Welcome screen : Configure | Create Desktop Entry.
- In the main menu: Tools | Create Desktop Entry.

You can choose to create the shortcut:

- For all of your computer users: In this case, the `jetbrains-phpstorm.desktop` entry file is created in `/usr/share/applications`.
- Only for yourself: the `jetbrains-phpstorm.desktop` entry file is created in `~/.gnome/apps` and `~/.local/share/applications`.

Importing PhpStorm usage preferences from a previous version

When you start PhpStorm for the first time, the Complete Installation dialog opens, and you have an option of importing PhpStorm usage preferences and license information from a previous version.

Select one of the following options and click OK.

- I want to import my settings from a previous version (<path>). If this option is present in the dialog, the directory that contains the PhpStorm settings and license information is found at its default location. (The corresponding path is shown within the option name in parentheses.) Most likely, this is the option that you want.
- I want to import my settings from a custom location. You may have the directory with the settings and license information available but not at its default location. If the option discussed above is missing, PhpStorm doesn't know where this directory is; the path initially shown in the field is just a guess, and there's no guarantee that the necessary folder is really there. To import the settings in this case, you can specify the folder with the settings (if you know where it is) , or an installation folder of the previous PhpStorm version. To do that, click ... and select the folder in the dialog that opens .
- I do not have a previous version of PhpStorm or I do not want to import my settings. If this is your case, select this option.

See also, Exporting and Importing Settings .

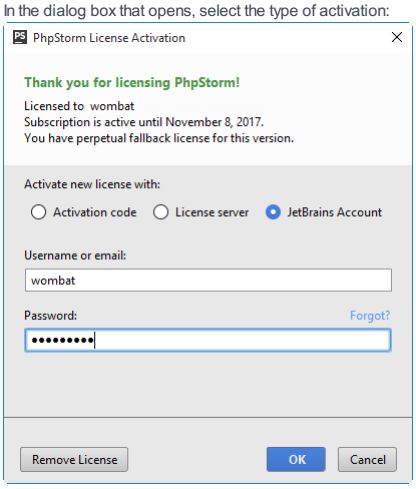
Registering PhpStorm

After you've performed installation and launching , you can register PhpStorm. This is how it's done:

To register PhpStorm

1. On the main menu, choose Help | Register...

2. In the dialog box that opens, select the type of activation:



   Depending on your selection, do one of the following:

   – Activation code: paste the activation code to the text area.

   – License server: specify the license server address, or click the button Discover server, if the address is not known.
   If you need to learn more, click the link More info... to open the License server documentation .

   – JetBrains Account: select this option if you have the JetBrains Account and wish to register with it.

3. In there are several licenses, choose the one you wish to use:

4. Click OK when ready.

The banner appears in the top-right corner of PhpStorm that says to whom is the copy of PhpStorm licensed.

Note that you can delete the unnecessary license. To do that, simpy click the button Remove License in the PhpStorm License Activation dialog.

Quick Start Guide

In this section:

- Before you start
- Initial configuration
- Set defaults from the Welcome screen
  - Change Settings/Preferences later
  - Difference between the L&F theme and the color scheme of the editor
- Work with projects
  - Create a project from downloaded sources
  - Create a project around sources from a Version Control Storage
  - Start a project from scratch
  - Populate your project
- Customize everything for your project!
  - VCS
  - Local history
  - File colors
  - IDE and Editor
    - Appearance
    - Editor
    - Keymap
    - External editor
- Background tasks
- Use macros
- Work with multiple files
- Find your way through
  - Search everywhere
  - Search for project elements
  - Search for code elements
  - IDE components
- Code with smart assistance
  - Edit and write code
  - Analyze your code
  - View documentation
  - Refactor safely
- View your changes instantly
- Run and debug your application
- Test your application
- That's about it!

Welcome to PhpStorm! This short guide aims to help you get a grip on the IDE.

Before you start

Tip Refer to the Guided Tour About PhpStorm User Interface to learn about the peculiarities of PhpStorm's UI.

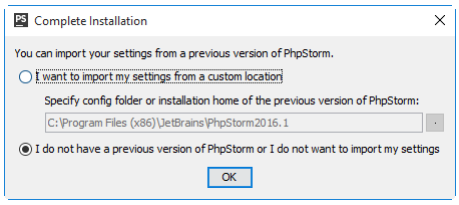
If you are going to run and debug an application directly on a remote host, the only thing you need is register access to this host in PhpStorm to enable synchronization with it.

If you plan to launch an application locally, you need a PHP engine installed and registered in PhpStorm and a Web server installed, configured, and integrated with PhpStorm . You can install these components separately or use an AMP package (Apache, MySQL, PHP solution stack). PHP engine version 5.4 and higher comes with a built-in web server, which is supported by PhpStorm.

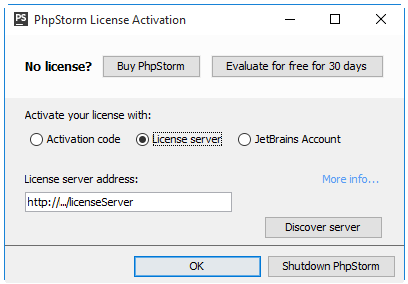
PhpStorm is cross-platform and works on Windows, Mac OS X, and Linux.

Initial configuration

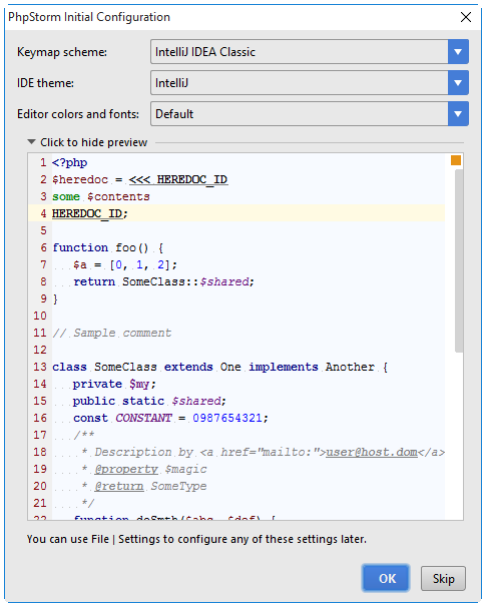
When you start PhpStorm for the very first time, you are asked whether you have a previous version and what to do with the settings. For the purposes of this tutorial, assume that there is no previous version, and click OK:



Next, you get the PhpStorm Licence Activation dialog, where you have to specify your particular license:



After you have specified your license, PhpStorm shows the initial setup dialog. Here you have to select the keymap and themes of the IDE and editor you want to use. Click the down arrow ▼ (Click to show preview) to assess what your choice will look like:



Note that PhpStorm has several pre-defined keymaps: for those who like Eclipse or Visual Studio, for Emacs fans, GNOME, KDE, and more. Explore the list of available keymaps in the Keymap page of Settings/Preferences dialog.

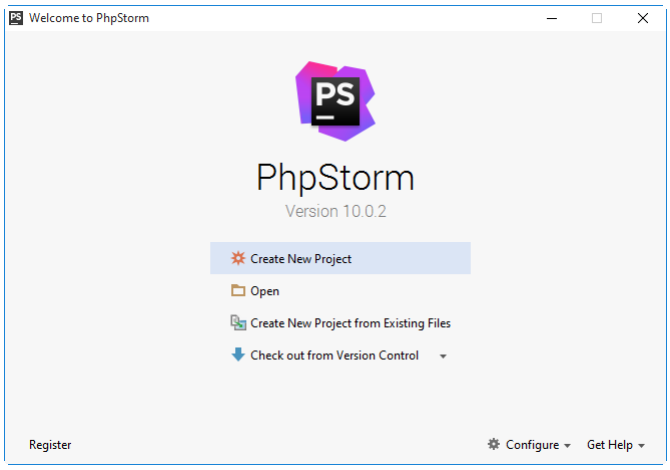
If later you decide that your initial choice of the keymap was wrong, you can always change your settings. To learn more about configuring the keymap, refer to the section Configuring Keyboard Shortcuts .

You can also change the IDE theme and the editor theme later .

There is also a IdeaVim plugin for PhpStorm that provides `vi/vim/gvim` emulation. Please follow

Set defaults from the Welcome screen

Again, if this is your very first start (or if you have no open project), you see the Welcome screen :



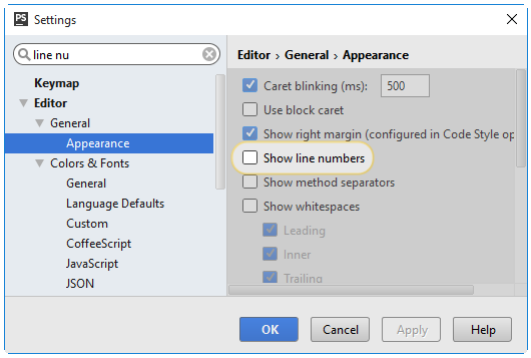
If you click Configure, PhpStorm shows the drop-down list that suggests you to configure settings/preferences, plugins, import and export settings, and check for updates. Choose Settings/Preferences and see the Settings/Preferences dialog . Look at this dialog - it has the header Default Settings/Preferences.

What does it mean? These settings will be used every time you create a new project.

The settings that pertain to a default project, are marked with the icon .

The Editor settings pertain to your working environment.

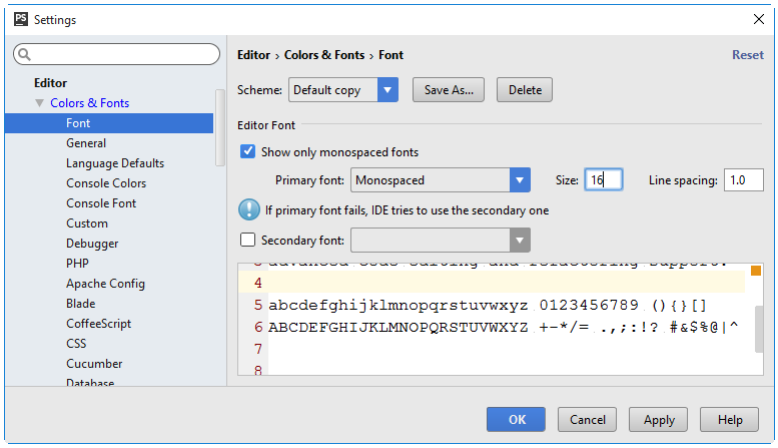
For example, you want to have line numbers always shown in the editor. In the Settings/Preferences dialog, expand the node Editor , and in the Appearance page, select the check box Show line numbers:



Suppose you want to use some particular color scheme for the editor, and you are not happy with the colors suggested by default. OK, select the base scheme, create a copy or it, and then change the colors (the schemes suggested by default are not editable).

It is also possible to set your preferred font size for the editor - this is done in the Font page under the Colors and Fonts node in the Settings/Preferences dialog. Again, you have to create a copy of the scheme first, if you haven't done it yet, and then specify the desired font size for the editor. This font size will apply to the current editor tabs and to all the newly opened editors. However, it does not affect the font size of the IDE components.

Note that you can always see the results of your experiments in the Preview pane:



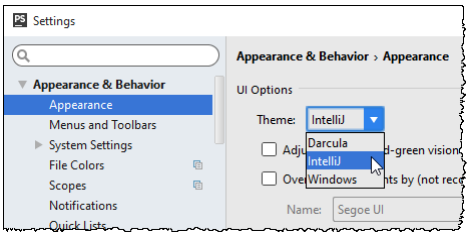
Change Settings/Preferences later

To change the look and feel of the IDE and the editor color scheme in a project, you have to open Settings/Preferences (press `Ctrl+Alt+S`, or click icon on the main toolbar).

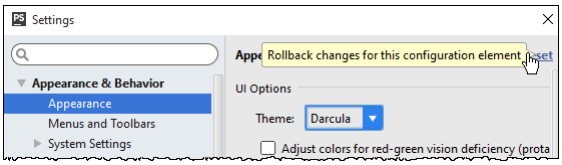
For example, you want to change Look and Feel (theme) of the IDE. Note that PhpStorm comes with several pre-defined themes. The set of available themes depends on your platform. You can find the theme of your choice in the Appearance page of the Settings/Preferences dialog.

This is how you can change the IDE theme:

1. On the main toolbar, click , or press `Ctrl+Alt+S`.
2. Expand the Appearance and Behavior node, and click Appearance.
3. On the Appearance page, select the theme you prefer from the drop-down list Theme:



Note the link Reset that appears on top of the dialog box - if you've changed your mind, click it to roll the changes back. Also note the button Apply that becomes enabled when you hover your mouse pointer over it:

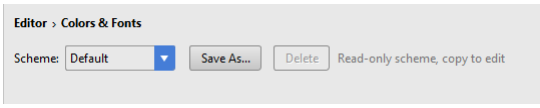


You can also change the other settings that define the look and feel of your PhpStorm, for example, the windows' behavior.

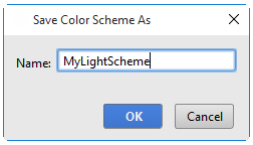
4. Apply the changes. You might be suggested to restart the IDE (for example, when you change theme to or from Darcula).

OK, the theme of PhpStorm has been changed, but you are still not happy with the appearance of your code in the editor. For example, you want another highlighting color for the current line. Let's do it.

1. Once more, on the main toolbar click to open the Settings/Preferences dialog, expand the node Editor , and then click Colors and Fonts .
   Note that you cannot change your pre-defined color scheme - you must create a copy of it first:

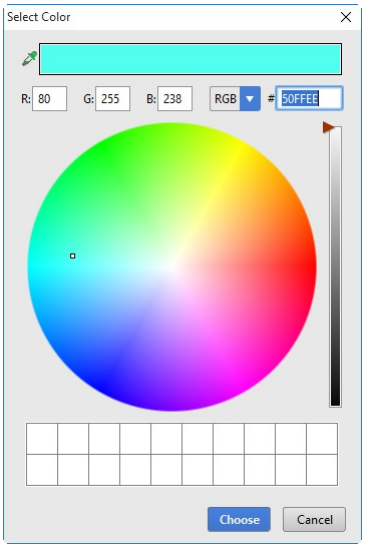


2. Click the button Save As..., and type the name of your new color scheme:



This new color scheme is editable, and you can proceed with changing it to your taste.

3. Under the Colors and Fonts node, click General, in the list of components, expand the Editor node, and locate the item Caret row.
4. Click the background color snippet to open the color picker :

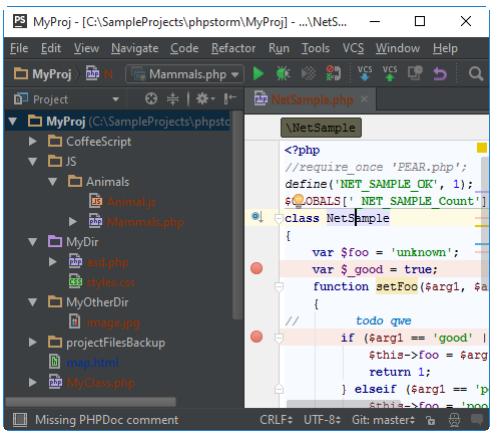


In this dialog box, pick the desired new color, and click Choose.

By the way, you have already noticed that some of the nodes are marked with . It means that the settings defined on these pages pertain to the current project. The pages without this icon contain settings that pertain to the entire working environment.

Difference between the L&F theme and the color scheme of the editor

Mind the difference between the IDE theme (look&feel) and the color scheme of the editor . The former relates to the entire IDE, while the latter relates to the editor only. It is quite possible to have a dark theme for the IDE (for example, Darcula), and one of the light color schemes for the editor (for example, Default). So doing, all the PhpStorm components will be rendered in one color scheme, and the editor pane in the different one:



Work with projects

Everything you do in PhpStorm, you do within the context of a project .

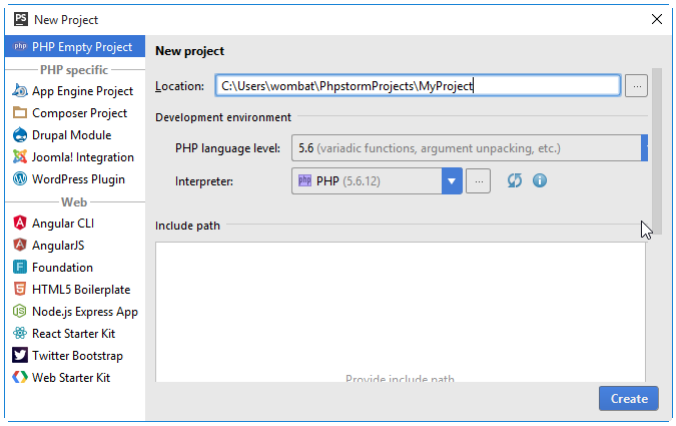
In PhpStorm, a project is a directory that contains source code, and a nested directory with the name `.idea` . The directory `.idea` is the place where PhpStorm stores all the necessary information - it's not visible in the Project view . From the user's point of view, a project is just a directory that contains some PHP code.

When you create a new project (File | New Project), or open an existing one (File | Open), PhpStorm suggests you to choose which way you want the project to be opened: in a new window, or in the same window after closing the previously opened project.

Refer to the section Creating and managing projects for details.

As an exercise, create a new empty project, where you can develop some pure PHP code. This case is discussed in detail in the tutorial Creating and Running Your First PHP Project .

By the way, PhpStorm suggests creating projects of the various types: PHP, Twitter Bootstrap, HTML5 Boilerplate, etc. You can explore the available types yourself, when creating a new project:



As you can see, PhpStorm supports all the major frameworks. For each of the supported project types, PhpStorm creates the corresponding file and directory structure, and all the necessary artifacts.

With PhpStorm, you can set up a project in the following ways:

- Download the sources from a remote host and arrange them in a PhpStorm project
- Clone a version control repository and create a project around the downloaded sources
- Start a project from scratch

Finally, when your project is ready, you can create new files and directories in it .

Create a project from downloaded sources

One of the most widely used workflows is updating an already existing application. In this case you need to download the application sources and arrange them in a PhpStorm project.

1. First of all, configure access to the remote host , where the sources are located (Tools | Deployment | Configuration).
2. Start the New Project from Existing Sources Wizard (File | New Project from Existing Files, then specify the method to access the remote host (FTP/SFTP/FTPS, local or mounted folder).

Refer to the section Creating a Project from Downloaded Files .

Create a project around sources from a Version Control Storage

You can also download sources from VCS storages and repositories, such as GitHub.

Choose VCS | Checkout from Version Control | <your vcs>). When asked by PhpStorm, type your credentials to access the storage.

Refer to the following sections:

- VCS-Specific Procedures
- Cloning a Repository from GitHub
- Setting up a Local Git Repository
- Checking out Files from SVN Repository

Start a project from scratch

You can start your project from scratch developing an application from the very beginning.

1. On the Welcome screen click Create New project, or on the main menu, choose File | New Project....
2. In the dialog box that opens, select the project type and enter the project properties such as project name, parent folder and project type.

Refer to the section Creating New Project From Scratch .

Populate your project

Adding new files or directories to your project is just a snap: select File | New... from the main menu or New from the context menu of the Project Tool Window .

Select the element of your choice and enter its name. PhpStorm does the rest: creates files by templates with some initial source code.

See Populating Projects for details.

Customize everything for your project!

Look at the main toolbar — there is a Settings button . Clicking this button opens the Settings/Preferences dialog box, where you can change your project structure, set up version control, and tweak your working environment to make the development process a real pleasure.

Some of the settings pertain to a particular project — for example, project interpreter, version control configuration, or file colors. They are marked . The others — like the Editor settings, keymaps, or live templates — pertain to your whole working environment, and thus can be configured even without an open project.

VCS

Surely, you keep your sources under a version control, right? Git? SVN? Mercurial? With PhpStorm, it's easy to set up, and again the IDE does a good job auto-detecting the VCS already used for existing projects.

But you can fine-tune — just click Version Control node in the Settings/Preferences dialog. By default, you will see the project root directory only, but you can break your application down into smaller chunks, and place virtually every directory under its own version control system.

In the Settings/Preferences dialog, you can also define some behaviors that are common for all version control systems: confirmation on creating or deleting files , tasks

performed in the background , ignoring unversioned files and more.

Refer to the sections under Version Control with PhpStorm for details.

Local history

In addition to a traditional version control system, you can use local history . With Local History, PhpStorm automatically tracks changes you make to the source code, results of refactoring, and state of the source code based on a set of predefined events (testing, deployment, commit or update). Local history is always enabled.

File colors

Your project might contain several sites, each one with its own set of files with the same names . When they are all opened in the editor, it's rather confusing... How can one tell which site they belong to? PhpStorm helps make them stand out by painting their editor tabs (Settings/Preferences|Appearance&Behavior|File Colors).

Break down your project into smaller chunks — scopes (for example, a scope per site), and select a color for each one.

Refer to the page Configuring Scopes and File Colors for details.

IDE and Editor

Working in an IDE actually means living in it, and it is quite natural to make your haunted place pleasant for your eyes and comfortable for your fingers. That's why PhpStorm makes it possible to choose a look and feel for the IDE and the editor, configure your preferred set of keyboard shortcuts (keymap), fine-tune the scrolling behavior, highlighting, and more.

It is recommended to familiarize yourself with the matter in the sections Project and IDE Settings and Configuring Settings .

All these settings are (again) configured in the Settings/Preferences dialog: click to open this dialog , and see the list of pages. Here are three pages that might be of interest for the starter: Appearance , Editor , and Keymap .

Appearance

Remember, you've initially selected the look and feel for your IDE on the first start? If you are not happy with the result, now it's time to change your mind. In the page Appearance page (Settings/Preferences | Appearance&Behavior | Appearance) you can select a "look and feel" of your PhpStorm installation. Just click the Look and feel drop-down, and select the scheme that suits you better. You don't need to close the Settings/Preferences dialog to observe the results of your experiments: click Apply, see what happens, and close the dialog when you are quite happy with what you've got.

Editor

The whole bunch of pages under the Editor node of the Settings/Preferences dialog helps you tune every aspect of the editor's behavior. Note that PhpStorm comes with the predefined color scheme, but if you want to make up something very personalized, you are welcome to do it: save the default scheme with a new name, and start changing its background, font, colors of syntactical elements, error highlighting, etc., and immediately observe the results in the preview pane.

Refer to the sections Configuring Project and IDE Settings and PhpStorm Editor. Guided Tour for details.

Keymap

The set of keyboard shortcuts you work with is one of your most intimate habits - your fingers "remember" certain combinations of keys, and changing this habit is rather painful. With this in mind, PhpStorm supplies you with a wide range of pre-defined keymaps (Keymap), for those who prefer Eclipse, or for those who've had long experience with Visual Studio... You can create your very own keymap on the base of an existing one.

Finally, there is the magic shortcut `Ctrl+Back Quote` that helps you switch between schemes (all of them — keymaps, colors, code styles, and L&F) without the Settings dialog (for Windows and Linux users only).

Refer to the section Configuring Keyboard Shortcuts for details.

External editor

Though you can choose any keymap that corresponds to your preferred editor (Emacs, Vim, TextMate, etc.), you might still want to open files in your preferred editor. You can easily do it by configuring an external tool. For example, you might want to open a current file in Emacs as an external tool.

Refer to Configuring Third-Party Tools for details.

Background tasks

Sometimes, when a long task is in progress, PhpStorm shows a Progress bar. You can bring such a task to the background, but still see how it goes on. Refer to page Working with Background Tasks for details.

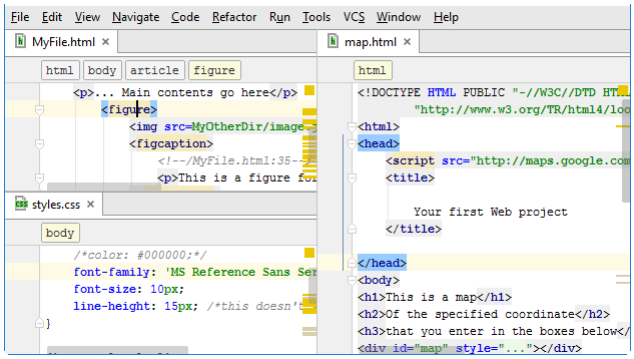
Use macros

It would be nice to have a chance to simplify your work with the editor. Using macros is just one of the possible ways. Refer to the product documentation for details.

Work with multiple files

As you might have noticed, PhpStorm opens each file in a separate tab in the editor.

PhpStorm provides a handy way to switch between the various tabs , allows pinning and unpinning each tab , placing editor tabs along any of the four borders of the window, splitting the editor tabs , and more.



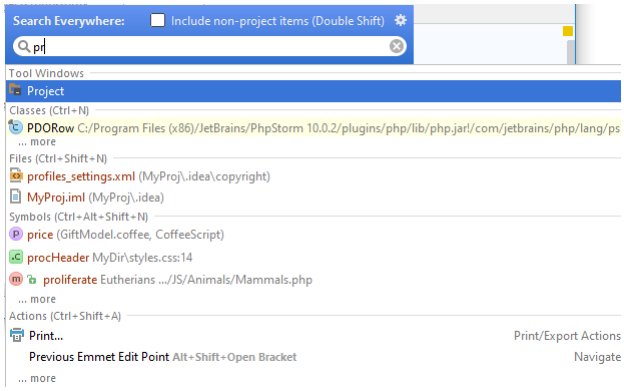
Refer to the documentation section Managing Editor Tabs for details.

Find your way through

Now when you know how to start working with PhpStorm, it's time to look at its navigation features. They will help you not to get lost and will also increase your productivity.

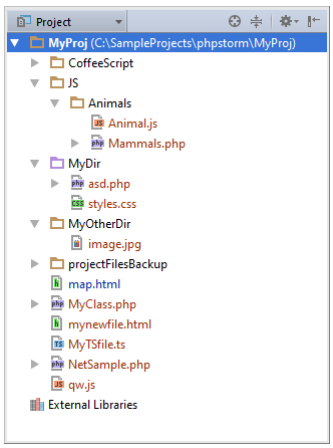
Search everywhere

Searching is for sure the most familiar and necessary option that is used for quick navigation. With PhpStorm you can search for any class, file, action or even an IDE setting with only one action . Double-press `Shift` or click the magnifier glass icon in the upper-right corner of the PhpStorm window:

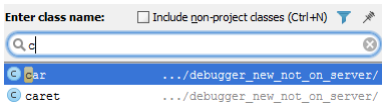


Search for project elements

The essential tool for project elements overview is the Project tool window (View | Tool Windows | Project tool window). It shows your project as a tree of files and directories. To access it, click the tool windows bar icon or press `Alt+1`:

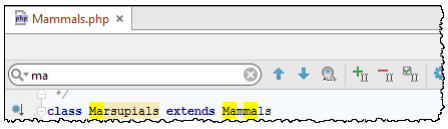


To quickly find an element by name and open it in the editor , use the navigation pop-up: press `Ctrl+N` (for a class), `Ctrl+Shift+N` (for a file), or `Ctrl+Shift+Alt+N` (for a symbol), and start typing the name you are looking for. The list of matching names shrinks as you type, but this is just one of the handy facilities: you can use the asterisk wildcard, all caps for CamelHumps, or spaces for snake_case names, slashes for nested folders and so on, and so forth .

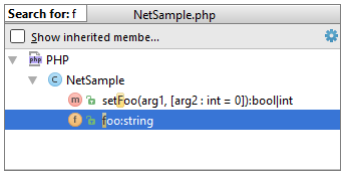


Search for code elements

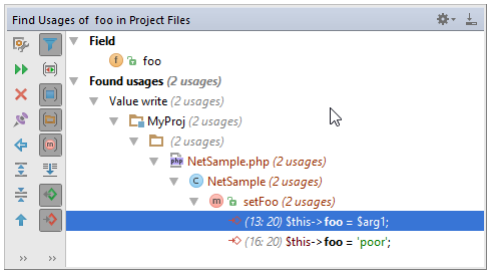
The basic way to navigate and search your source code is `Ctrl+F`. Start typing your search string, and get to its occurrences in the current file:



The ultimate way to navigate around the current file is to use the File Structure View (`Alt+7` or `Ctrl+F12`) and search through it:



Besides that, PhpStorm suggests a more sophisticated approach, namely search for usages . For example, if you want to navigate from a symbol to one of its usages within your application, press `Alt+F7`, or choose Find Usages on its context menu, and observe the results in the Find Tool Window :



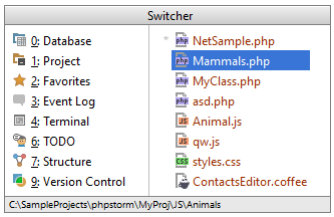
Use "Go to" actions to navigate to a declaration (`Ctrl+B`), implementations (`Ctrl+Alt+B`), type declaration (`Ctrl+Shift+B`), or super method (`Ctrl+U`).

Refer to the sections Searching Through the Source Code and Navigating Through the Source Code for details.

IDE components

There are many ways to navigate across the IDE . You may not remember all the actions and menu commands the first time around, and that's OK. Don't worry! Any time you need any IDE action, you can find it in Help | Find Action... (`Ctrl+Shift+A`) or use Search Everywhere . So, let's briefly outline some of the main commands.

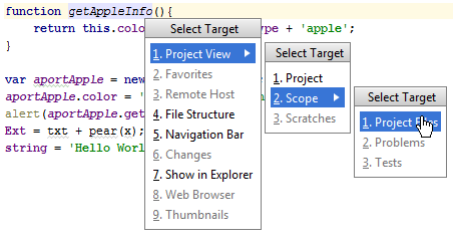
Let's start with the switcher : press `Ctrl+Tab` to show the switcher, which is a list of the PhpStorm's tool windows and open files, and then, keeping the `Ctrl` key pressed, use `Tab` or arrow keys to scroll to the component you want to go to:



Press `Ctrl+E` to see the list of recent files. Press `Ctrl+Shift+E` to see the list of recently edited ones.

If you have several IDE windows opened, use the `Ctrl+Alt+Open Bracket` and `Ctrl+Alt+Close Bracket` shortcuts to switch between them.

If you select a file in one of the IDE components and want to view it in another one (the editor, Project view, Navigation bar, or a changelist), then use Select Target (`Alt+F1`):



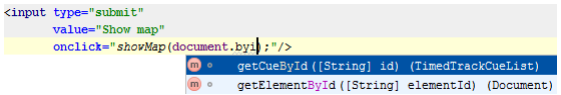
Finally, pressing `Escape` will bring you back to the editor, wherever you are. `Ctrl+Shift+F12` will hide all tool windows and leave only the editor.

Code with smart assistance

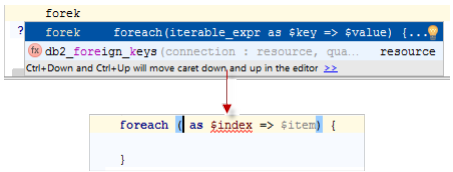
PhpStorm is famous for its smart code analysis features. Whatever you do to develop your source code, it is always at hand, helping you create error-free applications with no time wasted.

Edit and write code

At every stage of development, in any files, try to use code completion (`Ctrl+Space`), which takes into account the current context. Depending on where you invoke this action, you can complete keywords or code blocks, infer types, complete methods and properties , and more:



Enjoy live templates/code snippets (`Ctrl+J`) or surround templates (`Ctrl+Alt+T`) to produce entire code constructs:



PhpStorm comes with a wide range of ready-to-use live templates, or snippets, which you can explore in the Live Templates page of the Settings/Preferences dialog. If you see that you are lacking something especially important for your development goals, extend this set of snippets with your own ones.

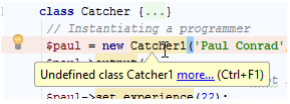
Don't also miss the possibility to surround with complete code constructs (choose Code | Surround With or press `Ctrl+Alt+T`).

PhpStorm helps you follow code style guides. All the new code is formatted automatically according to the code style settings (Settings/Preferences | Editor | Code Style). Nevertheless you may need to format some part of the code or apply changes made in code style settings. To do that, use Code | Reformat Code... (`Ctrl+Alt+L`).

Analyze your code

We, developers, spend only 20% of our time writing new code, but up to 80% improving the existing code. To help you avoid errors and correct them faster, PhpStorm gives you numerous hints and prompts based on the best development practices.

First, as you type, you immediately have all syntactical errors underlined with a red wavy line. If you place the caret at an error location, you will see a brief description of the problem at the tooltip, and also in the left side of the Status Bar . Besides that, you see red stripes along the validation bar, or the marker bar . If you hover your mouse pointer over such a stripe, you see the error description at the tooltip.

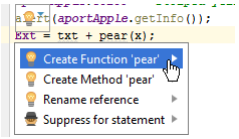


Tip Mind the marker bar on the right as you code. Red colored stripes denote errors, yellow stripes are warnings, and blue stripes stand for TODOs. Simply click them to navigate to an error, warning or a TODO note. A summary of the inspection results for the current file is shown in the colored indicator on top. Make it green!

– You can use Hector icon in the bottom right corner to configure inspections and set up the highlighting level for your files . By default, Hector never sleeps.

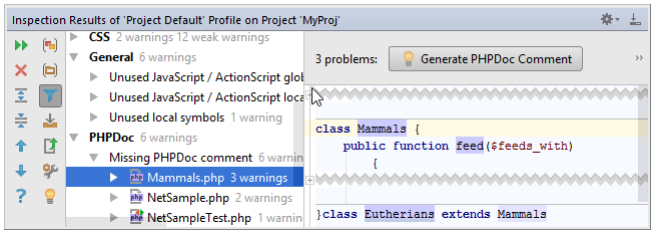
If you are sure of yourself, turn him off, and you will get neither warnings, nor suggestions for improvements or error messages.

As you type your code, PhpStorm, like a diligent pair programmer, keeps a close eye on what you are currently doing. Moreover, the IDE comes up with smart suggestions, also called intention actions , which are marked with the or icons. Click such an icon, or press `Alt+Enter` to auto-create a new method that you have not yet declared, fix an if/else statement, and much more:

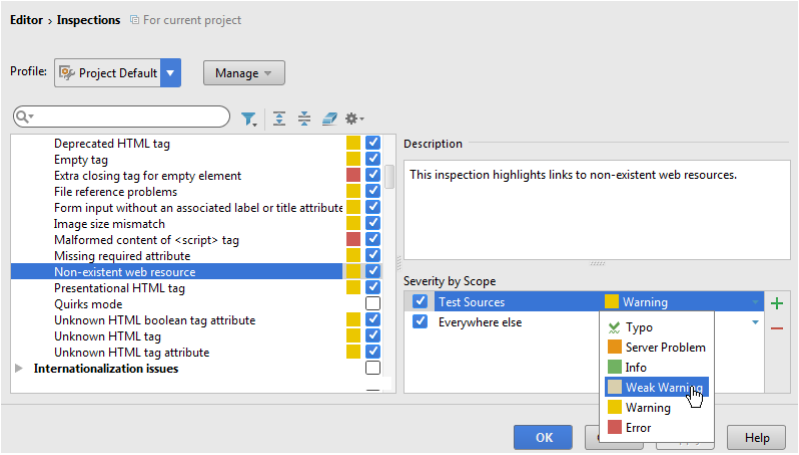


The next level is static code analysis, or code inspections : your code is analyzed without actually executing it. Actually, PhpStorm inspects code in the current file on-the-fly, and shows inspection results in the marker bar as colored stripes. If you see that the right side of your IDE frame is bright with red stripes, beware — it means that your code contains serious errors. Less important things, recommendations to improve code, or warnings, are displayed as yellow stripes. Information for the current file is summarized in a colored indicator on top of the marker bar, which works as traffic lights: green means that everything is fine; at least one problem turns the indicator yellow or red.

Sometimes you need to look deeper into your code. In this case, you have to inspect a whole project, or any of its parts (Code | Inspect Code) and explore the results in the Inspection Tool Window :



PhpStorm comes with a wide range of predefined inspections. All of them have a description and can be switched on or off, or configured as necessary in the Inspections page of the Settings/Preferences dialog:

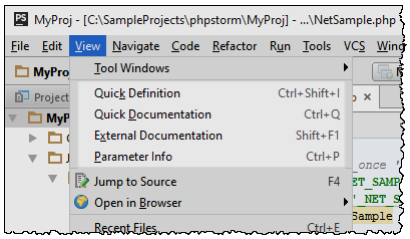


To help you make your source code less tangled and more readable, PhpStorm offers searching for duplicates in the whole project or its part (Code | Locate Duplicates...). Take a look at the detected repetitive fragments in the Duplicates tool window and decide what to do with them.

Read more in the sections Code Inspection , Intention Actions , and Analyzing Applications .

View documentation

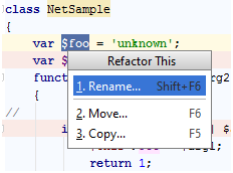
PhpStorm makes it possible to view existing documentation for the symbols at caret. There are several possible ways to do it:



Refer to the product documentation Viewing reference information .

Refactor safely

After inspecting your code, you may be interested in code refactorings provided by PhpStorm. The IDE helps you rename safely, extract methods, superclasses and modules, introduce variables, constants, parameters, and more. Use Refactor | Refactor This... (`Ctrl+Shift+Alt+T`) to view all refactorings available for the current code fragment.

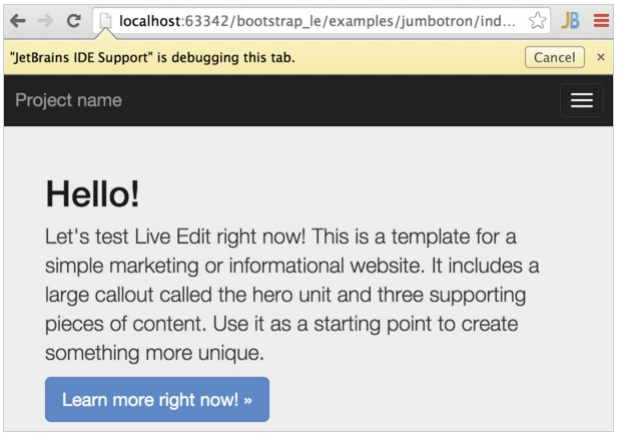


View your changes instantly

You can open an HTML, JavaScript, or CSS file in the browser and monitor how the changes you make to it in the editor are rendered without refreshing the page and even without leaving PhpStorm. By the way, completion look-up is also live. As you browse through the suggestion list, the page in the browser changes its appearance just as if you had already accepted the current suggestion. Currently Live Edit is supported only in Google Chrome, Chrome Canary, and Dartium.

There are some prerequisites for working with Live Edit. The Live Edit plugin should be installed in File | Settings | Plugins: click Install JetBrains Plugins, choose Live Edit and install this plugin. Also, make sure that the JetBrains Chrome Extension is installed on your Chrome browser.

The Chrome extension will be installed automatically if the Chrome Browser is not running during the first launch of the IDE with the Live Edit plugin installed. The Google Chrome Extension JetBrains IDE Support is available in Chrome Web Store .



For Live Edit to work correctly, you should run a JavaScript Debug Session or Node.js Debug Configuration . In order to start a JavaScript Debug Session, a run/debug configuration has to be created first. This can be done automatically using the Debug file_name context menu, or manually, if you want to specify additional options. Alternatively, open the page in Chrome by selecting Open in Browser from the context menu of an `html` file. When the page opens, select Inspect in PhpStorm from the context menu.

You can also refresh the page and view the changes without leaving PhpStorm, just choose Run | Reload in Browser on the main menu, when the JavaScript Debug Session is active.

See the tutorial Live Edit in PhpStorm and Live Editing of HTML, CSS, and JavaScript for details.

Run and debug your application

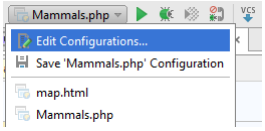
The easiest way to debug your web application is initiating a debugging session from the browser and working in the zero-configuration debugging mode , provided that you have created the corresponding debugger cookies and enabled control over debugging through them. We recommend starting debugging using this method. To connect to the running session, click the Start Listen PHP Debug Connections button on the toolbar. Refer to the tutorial for more information.

No preliminary steps are required if you are going to run and debug an application directly on a remote host. The only thing you need is register access to this host in PhpStorm to enable synchronization.

If you are going to run and debug an application on your computer, you need to configure local PHP Development Environment .

With PhpStorm, you can run entire PHP applications as well as a particular class or file. To run a class or a file, open it in the Editor or select it in the Project view, then choose Run <file name> on the context menu of the selection or just press `Ctrl+Shift+F10` .

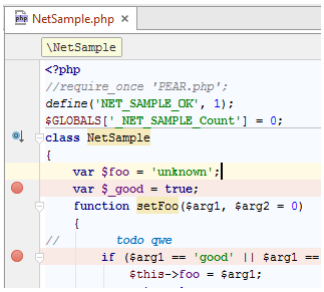
To launch an entire application you need a special profile, or a run/debug configuration , which represents a set of run/debug start-up properties. Run configurations are created in the Run/Debug Configurations dialog .



Depending on where you want your application to run and where you want to view its results, choose the configuration type:

- PHP Web Application to view application output in a browser.
- PHP Script to view the application output in the Run tool window.
- Built-in Web server .

Your application or script runs into a run-time error? To find out its origin, you will have to do some debugging. It starts with placing breakpoints (just click the left gutter of the line where you want the breakpoint to appear), at which program execution will be suspended, so you can explore program data:



PhpStorm supports integration with Xdebug and Zend Debugger. Before you start a debugging session, you need to download, install and configure one of these tools.

To initiate a debugging session from PhpStorm, you will need a debug configuration:

- PHP Web Application for entire applications.
- PHP HTTP Request for separate pages, for example, when you need to “come” to a specific page with certain data.

To start debugging, click the Debug button on the toolbar or press `Shift+F9`:

And last but not the least, you can also debug JavaScript in Google Chrome and Firefox right from PhpStorm, which is delivered with a built-in debugger . Create debug configuration of the type JavaScript Debug and click the Debug button on the toolbar.

Refer to the PhpStorm documentation for details:

- Zero-Configuration Debugging
- Configuring PHP Development Environment
- Running PHP Applications
- Debugging PHP Applications
- Running and Debugging JavaScript

Test your application

It is a good idea to test your applications, and PhpStorm helps doing it as simple as possible.

PhpStorm supports unit testing of PHP applications through integration with the PHPUnit tool. Before you start unit testing:

- Install and configure this tool depending on the operating system you use and your system settings.
- Integrate the tools with PhpStorm on the PHP page of the Settings/Preferences dialog.

To generate a stub of a test class, select the PHP class to create unit tests for and choose Generate PHPUnit Test.

Appoint the tests to run in a run configuration of the type Run/Debug Configuration: PHPUnit or Run/Debug Configuration: PHPUnit by HTTP , depending on whether you want to run tests locally or on the server. To run the tests, select the relevant configuration and click the Run button on the toolbar, or just press `Shift+F10`.

To start monitoring code coverage, select the relevant configuration and click the Run with coverage button on the toolbar.

That's about it!

Here we have given a very concise overview of some vital PhpStorm facilities, just to give you a quick start. There are numerous important features that can help make your life easier and your source code nicer and cleaner.

Enjoy developing with PhpStorm!

Keyboard Shortcuts You Cannot Miss

PhpStorm, as a keyboard-centric IDE, suggests keyboard shortcuts for most of its actions. In this topic, you can find a short list of the most indispensable of them, to make your first steps with PhpStorm easy.

> **Tip** PhpStorm also provides the default keymap reference in the `pdf` format. To view the keymap reference, choose Help | Keymap Reference from the main menu.

See the detailed list of default keyboard shortcuts in the Keyboard Shortcuts Reference and learn how to customize your preferred keymap in the Configuring Keyboard Shortcuts section.

Action	Shortcut
Find action by name	`Ctrl+Shift+A`
Show the list of available intention actions .	`Alt+Enter`
Switch between views (Project , Structure , etc.).	`Alt+F1`
Switch between the tool windows and files opened in the editor.	`Ctrl+Tab`
Show the Navigation bar .	`Alt+Home`
Insert a live template .	`Ctrl+J`
Surround with a live template .	`Ctrl+Alt+J`
Edit an item from the Project or another tree view .	`F4`
Comment or uncomment a line or fragment of code with the line or block comment.	`Ctrl+Slash` `Ctrl+Shift+Slash`
Find class or file by name .	`Ctrl+N` `Ctrl+Shift+N`
Duplicate the current line or selection .	`Ctrl+D`
Incremental expression selection .	`Ctrl+W` and `Ctrl+Shift+W`
Find/replace text string in the current file .	`Ctrl+F` `Ctrl+R`
Find/replace text in the project or in the specified directory	`Ctrl+Shift+F` `Ctrl+Shift+R`
Search everywhere.	Double-press `Shift`
Quick view the usages of the selected symbol .	`Ctrl+Shift+F7`
Expand or collapse a code fragment in the editor .	`Ctrl+NumPad Plus` `Ctrl+NumPad -`
Invoke code completion .	`Ctrl+Space`
Show the list of available refactorings (Refactor This).	`Ctrl+Shift+Alt+T`

The complete keymap reference is available from the main menu (Help | Keymap Reference).

First Steps

This part contains the basic tutorials:

- Create and Run Your First PHP Project
- Create and Run Your First Web Project

Create and Run Your First PHP Project

To get familiar with PhpStorm it is recommended that you create your first project from scratch and implement the very basic functionality in it.

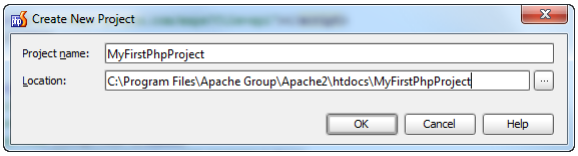
Creating a living PHP application that enables you to view results in a browser of your choice requires the following preconditions to be fulfilled:

- A local Web server is installed, configured, and running on your computer or access to a remote server is configured.
  This topic supposes that the local Apache HTTP server is used and its root directory is `.\htdocs`.
- The PHP engine is installed and configured.

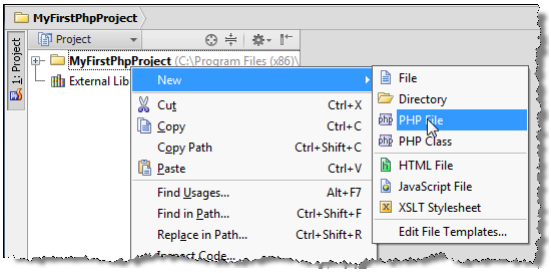
You can install each component separately and then configure the environment. Alternatively, install an AMP package compatible with the operating system you use to have the components and the entire environment configured without any steps from your side.

Creating and Run Your First PHP Project

1. Create a project. For that, choose File | New | Project on the main menu.
2. In the Create New Project dialog box that opens specify the name of the project in the Name text box, for example, type `MyFirstPhpProject`.
3. Click the Browse button ... next to the Location text box.
4. In the dialog that opens select the path to the `.\htdocs` folder and click OK. PhpStorm composes the path to the project folder as follows:

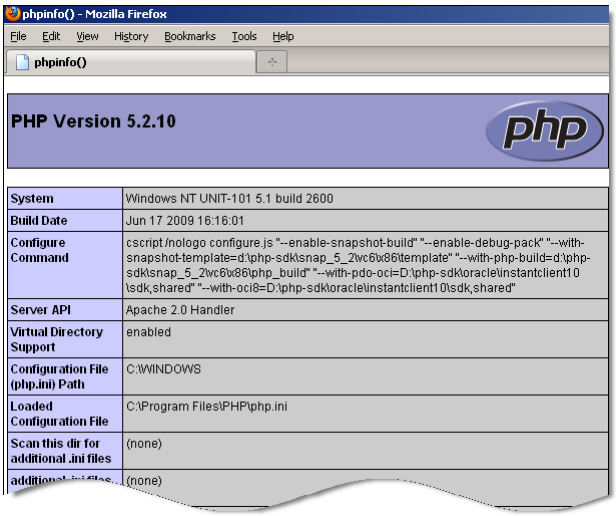


5. Create a PHP file. To do that, right-click the project directory in the Project tool window, point to New on the context menu, and choose PHP File.



6. In the New PHP File dialog box that opens type MyFile and click OK. PhpStorm will create the stub file for you and open it in the dedicated editor tab.
7. Type this sample code:
   `echo phpinfo();`
8. To save the file, choose File | Save All or press `Ctrl+S`.
9. Open a Web browser and type the following URL address:
   `http://localhost:<port>/MyFirstPhpProject/MyFile.php`

   The PHP Information page opens displaying the PHP engine configuration settings on your computer.

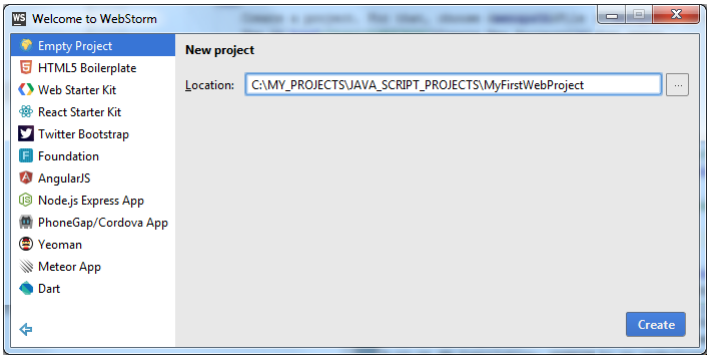


Create and Run Your First Web Project

To get familiar with PhpStorm it is recommended that you create your first project from scratch and implement the very basic functionality in it.

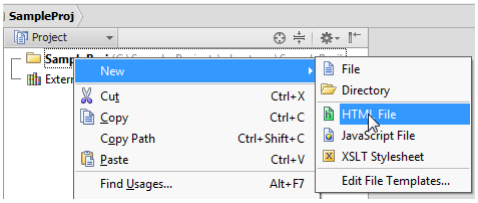
To create and run your first Web project:

1. Create a project. For that, choose File | New Project on the main menu. The Create New Project dialog box that opens.
2. In the left-hand pane, choose Empty Project.
3. In the right-hand pane, specify the path to the folder to create the project in, for example, C:\MY_PROJECTS\JAVA_SCRIPT_PROJECTS\MyFirstWebProject, in the Location text box. Type the path manually or click the Browse button ... next to the text box. Then select the desired folder in the dialog that opens . PhpStorm composes the path to the project folder as follows:



Click Create, when ready.

4. Create an HTML file. To do that, right-click the project directory in the Project tool window , point to New on the context menu, and choose HTML File.



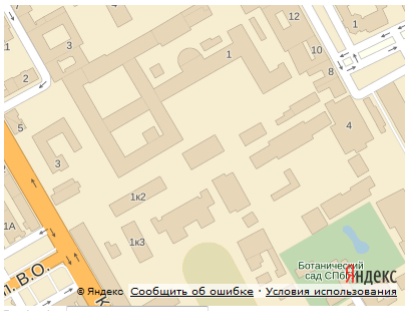
5. In the New HTML dialog box that opens, type MyFile and click OK. PhpStorm will create the stub file for you and open it in the dedicated editor tab.
6. Type this sample code inside the `<html />` tag:

```
<html>
<head>
<script src="http://api-maps.yandex.ru/2.0/?load=package.full&lang=ru-RU"></script>
<title>Your first
Web project</title>
</head>
<body>
<div id="map"
style="width: 400px; height: 300px"></div>
<label for="latitude">Latitude:</label>
<input type="text"
id="latitude" value="59.942402"/><br/>
<label for="longitude">Longitude:</label>
<input type="text"
id="longitude" value="30.293661"/><br/>
<input type="submit"
value="Show map" onclick=" showMap(document.getElementById('latitude').value,document.getElementById('longitude').value);
<script type="text/javascript">
function showMap (latitude, longitude) {
var myMap = new ymaps.Map('map',{
center:[latitude,
longitude],
zoom:16
});
}
</script>
</body>
</html>
```

Tip It is recommended that you type this code manually to experience coding assistance provided by PhpStorm.

7. Save the file by choosing File | Save All or pressing `Ctrl+S`.
8. Run your application. Do one of the following:
   - Choose View | Open in Browser, then select the desired browser from the list.
   - Click the desired browser on the browser toolbar:

9. In the page that opens in the browser, click the Show Map button. The Yandex map, that is displayed, shows the location of the IntelliJLabs office in Saint-Petersburg.



Latitude: 59.942402

Longitude: 30.293661

Show map

How to

This part provides descriptions of the platform and language-specific procedures.

- General Guidelines
- Language and Framework - Specific Guidelines

General Guidelines

This part provides descriptions of procedures required to fulfil the platform tasks with PhpStorm.

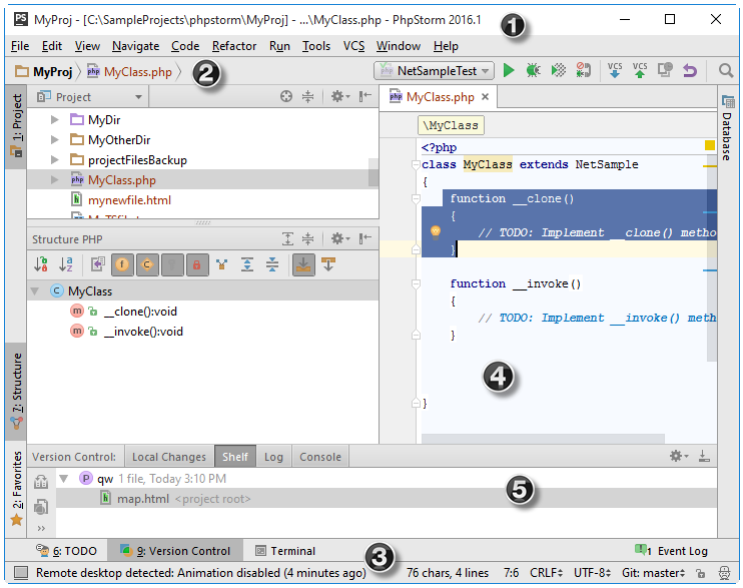
- Guided Tour Around PhpStorm User Interface
- PhpStorm Editor
- PhpStorm Tool Windows
- Configuring Project and IDE Settings
- Configuring Project Structure
- Creating and Managing Projects
- File and Code Templates
- Live Templates
- Populating Projects
- Generating Code
- Auto-Completing Code
- Creating and Optimizing Imports
- Code Inspection
- Intention Actions
- Viewing Pages with Web Contents
- Viewing Reference Information
- Navigating Through the Source Code
- Searching Through the Source Code
- Refactoring Source Code
- Analyzing Applications
- Working with Run/Debug Configurations
- Running
- Debugging
- Testing
- Code Coverage
- Working with Web Servers: Copying Files
- Using Local History
- Version Control with PhpStorm
- Managing Plugins
- Managing Tasks and Context
- Using Command Line Tools
- Using File Watchers
- Working with Embedded Local Terminal
- Running SSH Terminal
- Working with PhpStorm Features from Command Line
- Working with Background Tasks
- Using Language Injections
- Comparing Files and Folders
- Select File or Folder
- Copy and Paste Between PhpStorm and Explorer/Finder

Guided Tour Around PhpStorm User Interface

This chapter gives you insight into how PhpStorm user interface is organized to help you find your way through your working environment.

> **Tip** This chapter outlines the default (out-of-the-box) IDE interface layout. Note that plugins and other add-ons you are installing may change the way your IDE looks and behaves, for example there can be extra command buttons or menu items appearing in uncommon locations.

When you first run PhpStorm, or have no open project, PhpStorm displays the Welcome screen , which enables quick access to the major entry points. When a project is opened, PhpStorm displays the main window. This window is made up of the logical areas, which are shown on the picture below, marked with number labels.



1. Menus and toolbars - the main menu and toolbars let you carry out various commands.
2. Navigation bar that helps navigate through the project and open files for editing.
3. The status bar - indicates the status of your project, the entire IDE, and shows various warning and information messages.
4. The editor - here you create and modify the code.
5. PhpStorm tool windows - secondary windows that provide access to various specific tasks (project management, search, running and debugging, integration with version control systems, etc.).

Welcome Screen

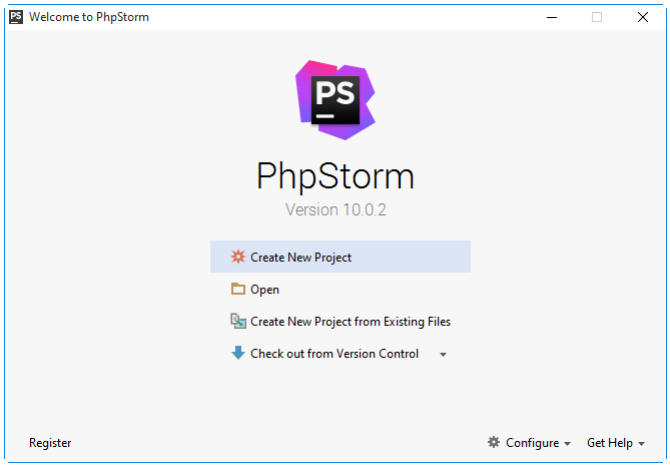
On this page:

- Overview
- Recent projects
- Quick start

Overview

PhpStorm displays the Welcome screen when no project is open. From this screen, you can quickly access the major starting points of PhpStorm. The Welcome screen appears when you close the current project in the only instance of PhpStorm. If you are working with multiple projects, usually closing a project results in closing the PhpStorm window in which it was running, except for the last project, closing this will show the Welcome screen.

The Welcome screen is divided into the following sections: Quick Start and Recent Projects (if any).

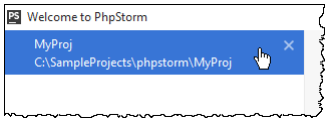


Tip Use the `Tab` key to navigate through the Welcome screen.

Recent projects

If appropriate, the left-hand pane shows a list of projects you've recently been working with. Click a project to reopen it.

To find a project of interest, start typing its name.



To delete a recent project from the list, follow these steps:

1. Use the `Tab` key to move focus into the list of Recent projects.
2. Use arrows keys to select the project you'd like to remove, or find it, as shows above.
3. Do one of the following:
   - Press Delete on your keyboard and confirm deletion in the Remove Recent Project dialog box that opens.
   - To remove the selected recent project silently, click × or choose Remove Selected from Welcome Screen on the context menu of the selection.

Quick start

Use the links of this section to create a new project , open an existing project, or check out a project from version control.

Also, use the drop-down arrows (▾) Configure to configure your working environment and default project , and Get Help to open help topics, tips of the day, and default keymap document.

Menus and Toolbars

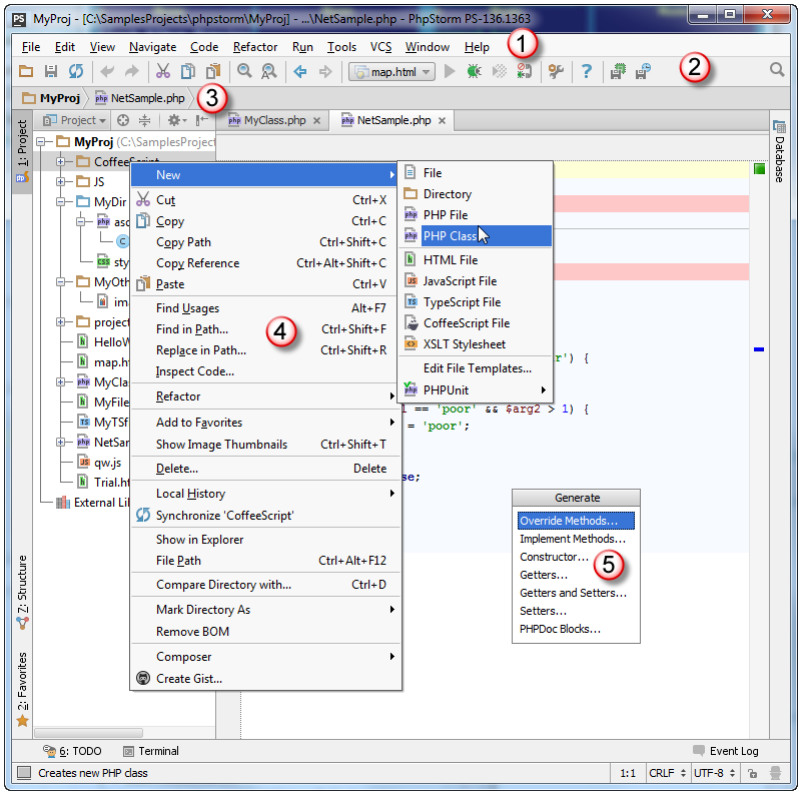
On this page:

- Overview
- Main elements of PhpStorm window
- Tips and tricks

Overview

PhpStorm menus and toolbars let you carry out various commands. The main menu and toolbar contain commands that affect the entire project or large portions of it. Additionally, context-sensitive pop-up menus let you perform the commands, which are specific to a part of a project, like source file, class, etc. Almost each of the commands has an associated keyboard shortcut to enable quicker access to it.

Use the check commands of the View menu to show or hide the main elements of the PhpStorm window. For example, if you want to show the main toolbar, make sure that the check command Toolbar is selected.



Main elements of PhpStorm window

1. Main menu

   The main menu contains commands for opening, creating projects, refactoring the code, running and debugging applications, keeping files under version control and more.

2. Main toolbar

   The main toolbar contains buttons that duplicate the essential commands for quicker access. You can hide the main toolbar, using the checked command on the toolbar context menu.

   By default, the main toolbar is hidden. To show it, select the check command View | Toolbar on the main menu.

3.Navigation bar

   Navigation bar is a quick alternative to the Project tool window.

4. Context menus

   These menus, available with right-click, contain commands applicable to the current context.

5. Pop-up menus

   These menus, available with `Alt+Insert`, contain commands applicable to the current context.

Tips and tricks

- Show or hide the main elements of PhpStorm UI using the View menu.
- Descriptions of the actions from all the menus and toolbar buttons are displayed in the left side of the Status bar.
- If you know which action you want to perform, but do not know where to find the appropriate command, just press `Ctrl+Shift+A`, type some part of the name of action you want to perform, and select the desired action from the suggestion list.

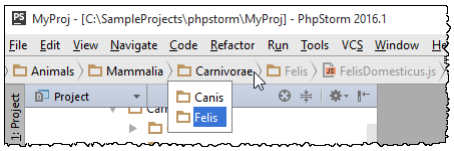
Navigation Bar

On this page:

- Introduction
- Toggling the Navigation Bar
- Tips and tricks

Introduction

Navigation Bar is a quick alternative to the Project view . Use this tool to navigate through the project and open files for editing .



Toggling the Navigation Bar

To show the Navigation Bar, do one of the following

- On the View menu, select the check command Navigation Bar.
- Press `Alt+Home`.

To hide the Navigation Bar

- On the View menu, clear the check command Navigation Bar.

Tips and tricks

Please note the following:

- When the main toolbar is hidden, the Navigation Bar shows the run/debug configuration selector, run and debug , version control buttons (if version control integration is enabled) and search everywhere magnifying glass .
- When the Navigation bar is hidden, press `Alt+Home` to open it floating.
- Pressing `Escape` returns focus to the editor.

Status Bar

On this page:

- Introduction
- Status Bar icons

Introduction

PhpStorm Status Bar indicates the current IDE state and lets you carry out certain environment maintenance tasks.

Status Bar icons

Icon	Description
	Click to toggle showing or hiding tool window bars. Also double press and hold ⌘ (for OS X) or Alt (for Windows or *NIX) to show hidden tool window bars. Refer to the procedure description .
Load offline inspection results	This section of the Status bar shows description of a command, currently selected on the main menu, context menu, or a toolbar.
	Click this icon to invoke the Background Tasks manager . Visibility of this icon in the Status bar depends on a launched background task.
48:82 48:82/17	The first two numbers denote the current caret position in the editor(line and column). If a selection is made in the editor, PhpStorm shows the length of the current selection after slash.
	This Lock icon indicates the read-only or writable attribute of the current file in the editor. To toggle the file attribute, click the Lock icon or use the File | Make File Writable/Read-only command on the main menu.
CRLF	View and change line endings of the current file in the editor.
windows-1251	View and change encoding of the current file in the editor.
	Click this icon to navigate to the pending source control changelists in the Incoming tab of the Version Control tool window.
	Hovering your mouse pointer over the icon shows the current code inspection profile at the tooltip. Clicking the Hector icon results in showing a dialog box with the following functions: – Highlighting level. Use the slider to change highlighting level for the current file, or configure inspection profile . Depending on the highlighting level selected by the slider, Hector keeps an eye on the code (Inspection level), turns half face (Syntax), or averts his face from the code (None). – Power Save Mode. Select this check box to minimize power consumption of your computer on account of eliminating the background operations. To indicate that the mode is on, Hector fades . When Power Save Mode is on, PhpStorm reduces its functionality to the one of a text editor, by not executing expensive background activities that drain laptop battery. These activities include error highlighting and on-the-fly inspections, autopopup code completion . You can also toggle Power Save Mode through the File | Power Save Mode command on the main menu.
	Indicates that there are unattended notifications. Click this icon to see the notification descriptions in the Event Log tool window.
	Alternatively, when the icon is empty, there are no new notifications.
	This blinking icon indicates that internal IDE errors have occurred. Click to view the error descriptions and submit reports.
118M of 494M	Shows the current heap level and memory usage. Visibility of this section in the Status bar is defined by the Show memory indicator check box in the Appearance page of the Settings/Preferences dialog. It is not shown by default. Click the memory indicator to run the garbage collector.

Note More icons appear in the Status Bar as you download and install plugins.

PhpStorm Viewing Modes

On this page:

- Basics
- Toggling full screen mode
- Toggling presentation mode
- Toggling distraction-free mode

Basics

PhpStorm provides special view modes:

- Full Screen mode allows you to use the entire screen for coding. This removes all menus from view, as well as the operation system controls. However, you can use context menus and keyboard shortcuts. The main menu is also available when you hover the mouse pointer over the top of the screen.
- Presentation mode - this mode is similar to the Full Screen mode, but it is designed for making presentations that involve coding with PhpStorm. In this mode, PhpStorm increases the font size and hides everything except the editor. If necessary, tool windows can be also displayed in this view using the corresponding items in the View | Tool Windows menu.
- Distraction-free mode shows no toolbars, no tool windows, no editor tabs; the code is center-aligned; etc.

Toggling full screen mode

Besides manipulating the tool windows (show/hide or resize them), PhpStorm makes it possible to maximize the entire product window, hiding the main menu.

To enter full screen mode

- On the main menu, choose View | Enter Full Screen.

To exit full screen mode

1. Hover your mouse pointer over the top-most line to reveal the main menu.
2. On the main menu, choose View | Exit Full Screen.

Toggling presentation mode

In the presentation mode, the editor occupies the entire screen, while all the other PhpStorm components are hidden.

Besides that, the font size in this mode is larger than usual. You can define font size for the presentation mode in the Appearance page of the Settings dialog.

To enter presentation mode

- On the main menu, choose View | Enter Presentation Mode.

To exit presentation mode

1. Hover your mouse pointer over the top-most line of the editor to reveal the main menu.
2. On the main menu, choose View | Exit Presentation Mode.

Toggling distraction-free mode

In the distraction-free mode, the editor occupies the entire PhpStorm frame, without any editor tabs and tool-window buttons. The code is center-aligned.

To enter distraction-free mode

- On the main menu, choose View | Enter Distraction-Free Mode.

To exit distraction-free mode

- On the main menu, choose View | Exit Distraction-Free Mode.

Tip These actions are not mapped to any shortcuts by default. You can create your own shortcuts as described in the section Configuring keyboard shortcuts .

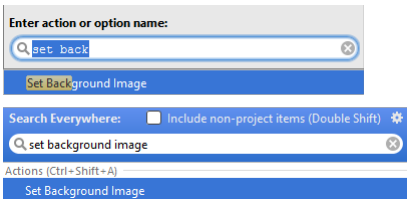
Setting Background Image

PhpStorm allows you to define any image as a background. So doing, you can set a background image for the current project only, or for any project you open or create anew.

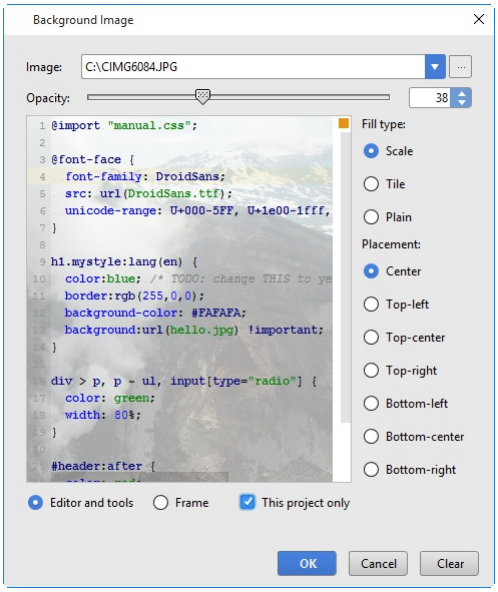
This feature has no keyboard shortcut (you can easily create a shortcut as described in the section Configuring Keyboard Shortcuts).

To set a background image

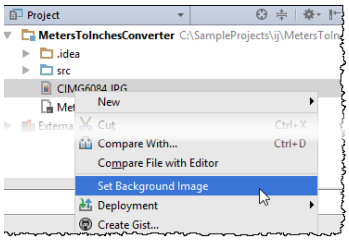
1. Do one of the following:
   - Press `Shift` twice (see Searching Everywhere).
   - Press `Ctrl+Shift+A` (see Navigating to Action).



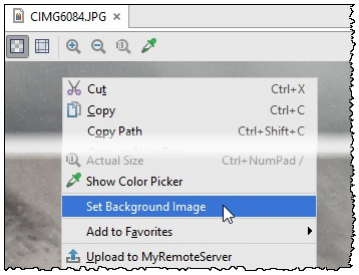
2. In the dialog box that opens, specify the image you want to use as the background, its opacity, filling and placement options. Besides that, you can choose to show background in the editor and tool windows, or in the PhpStorm frame.
   Also, select the check box This project only to show background in the current project, and ignore this background in the other projects.



If an image is already selected in a PhpStorm project, this action is also available from the context menu of the Project tool window:



and in the image editor



PhpStorm Editor

In this section:

- PhpStorm Editor
  - Basics
  - Active editor
  - Editor's areas
- Basic Editing Procedures
- Advanced Editing Procedures
- Managing Editor Tabs
- Using TODO

Basics

PhpStorm editor is a powerful tool for creating and modifying source code. As any other IDE editor, it supports basic features like bookmarks , breakpoints , syntax highlighting , code completion , zooming , folding code blocks , etc. There are, however, plenty of advanced features like macros , highlighted TODO items , code analysis , intention actions , intelligent and fast navigation , and a lot more.

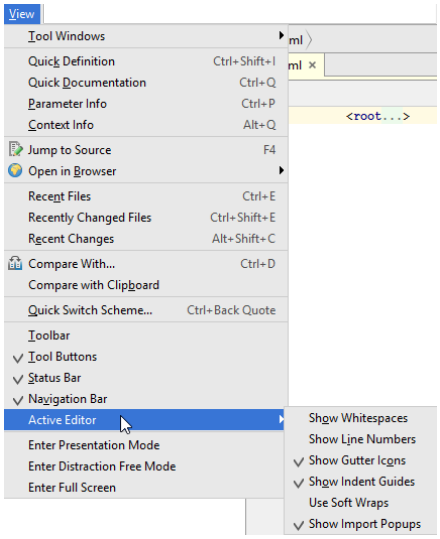
To configure your editing environment, use the Editor settings page and its child pages. There is also a Quick Switch Scheme command that lets you change color schemes, themes, keymaps, etc. with a couple of keystrokes.

The editor is tab-based. All operations with the editor tabs are available from the context menu of a tab , or from Window | Editor tabs node of the main menu .

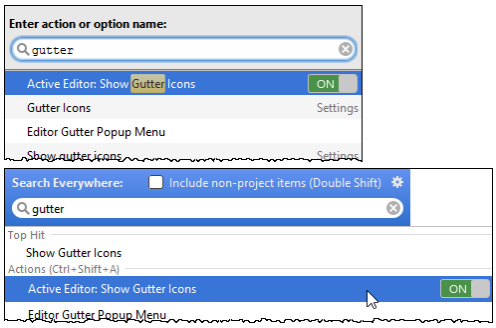
Active editor

Tip You always return the focus to the active editor from any tool window by pressing the Escape key.

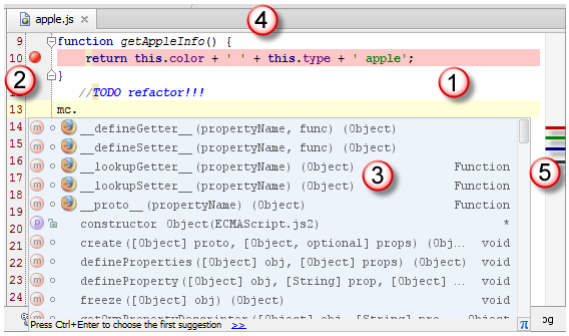
When you open a file for editing , it opens in its own tab. The editor you are currently working in, is the active editor. You can change behavior of the active editor using the commands under View | Active Editor node of the main menu:



Alternatively, you can invoke the commands related to the active editor, from Find Action or Search Everywhere :



Editor's areas



1. Editor area

   Use this area to type and edit your source code. The editor suggests numerous coding assistance facilities. Refer to the sections under this node, and to Basic Editing Procedures and Advanced Editing Procedures for details.

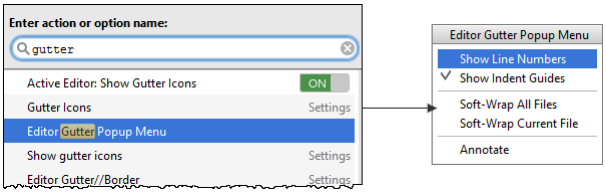
2. Gutter area

   The left gutter provides additional information about your code and displays the various icons that identify the code structure, bookmarks, breakpoints, scope indicators, change markers and the code folding lines that let you hide arbitrary code blocks.

   You can change the behavior of the left gutter.

   For example, it's possible to make the left gutter thinner by hiding the gutter icons. This is done either for the active editor , or for all the newly created editors .

   To change the behavior of the left gutters, use either the Appearance page of the editor settings, or the Editor Gutter Popup Menu:



   By default, this command is not mapped to any keyboard shortcut. You can create your own shortcut as described in the section Configuring Keyboard Shortcuts .

3. Smart completion pop-up

   This is one of the key editing assistance features that suggests method names, functions, tags and other keywords you are typing.

4. Document tabs

   Enable quick navigation across the multiple documents you are working on. Clicking a tab brings its contents to front and makes it available for editing in the active editor.

   To move between the tabs, use the keyboard shortcuts `Alt+Right` or `Alt+Left`.

   Clicking a tab while the `Ctrl` / `⌘` key is pressed, allows navigating to any part of the file path, through opening it in an external browser.

   Context menu of a tab provides all commands applicable to a file opened in the editor, for example:

   - Close one or more tabs.
   - Pin active tab.
   - Split and unsplit tabs.
   - Manage groups of tabs.
   - Navigate between tabs.
   - Add to Favorites.
   - Move to a changelist.
   - Run , or debug in the active editor.
   - Perform local history and version control commands.
   - Perform commands of your own tools .

   By default, the tabs appear on top of the editor, but you can change their location as described in the section Changing Placement of the Editor Tab Headers .

5. Validation side bar / marker bar

   This is the bar to the right from the editing area, showing the green, red or yellow box on its top depending on whether your code is okay, or contains errors or warnings. This bar also displays active red, yellow, white, green and blue navigation stripes that let you jump exactly to the erroneous code, changed lines, search results, or TODO items.

Basic Editing Procedures

This section describes how to perform the most common editing tasks:

- Opening and Reopening Files in the Editor
- Closing Files in the Editor
- Selecting Text in the Editor
- Cutting, Copying and Pasting
- Copy and Paste Between PhpStorm and Explorer/Finder
- Using Drag-and-Drop in the Editor
- Commenting and Uncommenting Blocks of Code
- Saving and Reverting Changes
- Undoing and Redoing Changes
- 'Lens' Mode
- Multicursor
- Scratches
- Zooming in the Editor

Opening and Reopening Files in the Editor

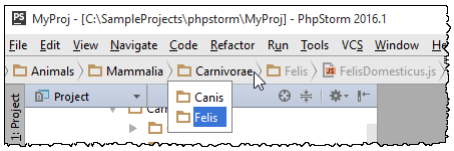
On this page:

- Opening files for editing
- Opening external files
- Reopening files
- Opening files in a separate window

Opening files for editing

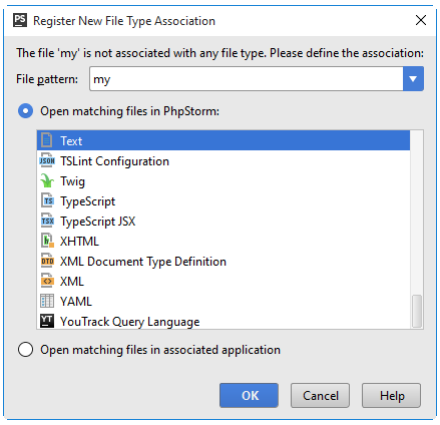
To open a file for editing

1. Do one of the following:
   - Double-click the desired file in one of the Tool Windows .
   - Select the desired file in one of the Tool Windows and press `F4`.
   - Select the desired file in the one of the Tool Windows and choose Jump to Source on the context menu.
   - Use the Navigate command for a Class, File, or Symbol.
   - Click the desired directory in the Navigation bar , and select file from the drop-down list:



2. If the file type is registered, the file opens silently in PhpStorm's editor.
   If the file type is registered under the category Files opened in associated applications, it will be opened in its associated application, rather than in the PhpStorm editor. By default, PhpStorm suggests a number of such file types, for example `.doc` , `.chm` , or `.pdf` .

   If the file type is unknown, PhpStorm suggests you to choose whether you want to register a new file type, or open such file in its associated application. Specify your choice in the Register New File Type Association dialog box:



   You can register the required file types on the File Types page of the Settings/Preferences dialog.

The maximum size of files parsed by PhpStorm is controlled by the `idea.max.intellisense.filesize` setting in `idea.properties` file.

The file `idea.properties` , located in the `bin` directory of the PhpStorm installation folder, should not be edited. Instead of editing the original `idea.properties` , create file `idea.properties` in the location specified below, open it for editing and add the required properties.

So, depending on your platform:

- For Windows: in `%USERPROFILE%\.PhpStorm XX`
- For *NIX: in `~/.PhpStorm XX`
- For OS X: in `~/Library/Preferences/PhpStorm XX`

Note that the larger the file is, the slower its editor works and the higher overall system memory requirements are.

Opening external files

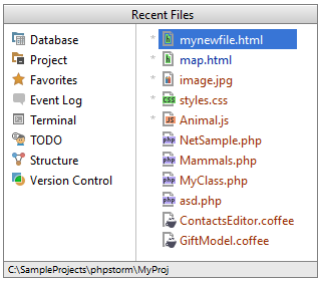
Do one of the following:

- Choose File | Open on the main menu and select the desired file in the dialog box that opens.
- Drag the required file from the Explorer (Windows), File Browser (Linux), or Finder and drop it to the editor. The file opens for editing in a new tab.

Reopening files

To reopen a file

- To open a recently opened file, choose View | Recent Files on the main menu or press `Ctrl+E`. Then select the desired file from the Recent Files pop-up window, that opens.



– To open a recently updated file, on the main menu, choose View | Recently Changed Files or press `Ctrl+Shift+E`. Then select the desired file from the Recently Edited Files pop-up window, that opens.

> **Tip** Use Recent files limit text box in the Editor settings page to define the maximum number of recent files.

Opening files in a separate window

To open a file in a separate PhpStorm window

Do one of the following:

– Drag and drop an editor tab outside of the current PhpStorm window.

– Press `Shift+F4` for a file selected in the Project tool window.

– `Shift+mouse double click` on a file name in the Project tool window.

Closing Files in the Editor

PhpStorm suggests several ways to close editor tabs.

To close a file in the editor, do one of the following

- On the main menu point to Window | Editor Tabs, choose one of the appropriate closing commands.

  Close

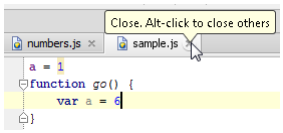
  Closes the file in the active tab.

  Close All

  Closes all editor tabs.

  Close Others

  Closes all tabs except the current one. The alternative way to close all other tabs lays with clicking the × button, while holding the `Alt` key pressed:



  Close Unmodified

  Closes all files that were not changed. This command is only available, when version control integration is enabled in project.

  Close All But Pinned

  Closes all files that were not pinned. This command appears, if there are pinned editor tabs.

- Right-click any editor tab, and choose same commands on the context menu.

- Point with your mouse cursor to a tab and click the middle mouse button.

- Point with your mouse cursor to a tab and click ×.

- Press `Ctrl+F4`.

Tip When you close modified files, PhpStorm preserves all changes in the current editing session. After reopening such files, the results of editing are restored.

Selecting Text in the Editor

On this page:

- Basics
- Selecting all text in the active editor tab
- Selecting with navigation keys
- Alternative ways to select code
- Extending selection
- Shrinking selection
- Multiselection
- Toggling between line and column selection modes
- Using smart selection
- Sticky selection
- Tips and tricks

Basics

The basic way to select a piece of text is to extend the selection with the mouse cursor. PhpStorm, as a keyboard-centric IDE, suggests to use navigation keys to make selections.

You can opt to select pieces of text by lines, or select rectangular fragments in the column mode, extend and shrink the selection, use multiselection, and sticky selection.

Selecting all text in the active editor tab

To select the entire text in the current editor tab, do one of the following:

- On the main menu, choose Edit | Select All.
- Press `Ctrl+A`.

Selecting with navigation keys

To select text from the current caret position to the beginning/end of the current word:

- `Ctrl+Shift+Left`, `Ctrl+Shift+Right`.

To select text from the caret position to the beginning/end of the current line:

- Double-click `Ctrl` and press `Home` / `End`

To select text from the current caret position to the top/bottom of the screen:

- `Ctrl+Shift+Page Up`, `Ctrl+Shift+Page Down`.

Alternative ways to select code

To make selection of a column of text, do one of the following:

- Keeping the `Alt` key pressed, drag your mouse pointer to select the desired area.
- Keeping the middle mouse button pressed, drag your mouse pointer to select the desired area.
- Press `Shift+Alt` and the middle mouse button. This is specially helpful, if you want to avoid dragging.

Extending selection

To extend selection from the word at caret to the piece of code the caret is contained in, do one of the following:

- On the main menu, choose Edit | Extend Selection
- Press `Ctrl+W` to select the word where the caret is currently located.
- Press `Ctrl+W` successively to extend selection to the next containing node (for example, an expression, a paired tag, an entire conditional block, a method body, a class, a group of vararg arguments, etc.)
  While extending selection, keep in mind that:
  - Pressing `Ctrl+W` successively in plain text or comments extends the selection first to the current sentence, then to the current paragraph.

Shrinking selection

To shrink selection in the reverse order (from the outermost container to the word where the caret currently resides), do one of the following:

- On the main menu, choose Edit | Shrink Selection
- Press `Ctrl+Shift+W`.

> **Tip** The selection extends or shrinks according to capitalization, if the Use "CamelHumps" words check box is selected on the Smart Keys page of the editor settings.
>
> If you want to make selection according to capitalization, using double-click, make sure that the check box Honor CamelHumps words... is selected on the General page of the editor settings.

Multiselection

PhpStorm supports selecting multiple text fragments. So doing, one can select multiple words, lines or rectangles.

To select multiple words, follow these steps

1. Do one of the following:
   - Press `Alt` and double-click the left mouse button.

```
<procedure title="To select multiple words" alternative-title="Using m
    <step...>
</procedure>
<anchor name="column_selection"/>
<procedure title="To toggle between the line and the column selection
<anchor name="altdrag"/>
<procedure title="To make selection in the Column Selection Mode"
```

– Press `Alt+J`, or select some text fragment. Then press `Alt+J` again, to find the matching piece of text.

2. After selection is complete, you can start editing all the fragments as if they were one.

To select multiple fragments of text, follow these steps

1. Press `Alt`

2. Drag the mouse pointer

```
s.css
1 body {
2     background: #ffffff;
3     color: #3f3f3f;
4     /*color: #000000;*/
5     font-family: 'MS Reference Sans Serif', verdana, arial, sans-serif;
6     font-size: 10px;
7     line-height: 15px; /*this doesn't work*/
8 }
```

To select multiple rectangular fragments of text, follow these steps

1. Press `Alt` (Windows or UNIX)/ (OS X)

2. Drag the mouse pointer

```
s.css
1 body {
2     background: #ffffff;
3     color: #3f3f3f;
4     /*color: #000000;*/
5     font-family: 'MS Reference Sans Serif', verdana, arial, sans-serif;
6     font-size: 10px;
7     line-height: 15px; /*this doesn't work*/
8 }
```

Refer to the section Multicursor for additional information.

Toggling between line and column selection modes

To toggle between the line and the column selection modes, do one of the following:

– On the main menu, choose Edit | Column Selection Mode.

– On the context menu of the editor, choose Column Selection Mode.

– Press `Shift+Alt+Insert`.

Using smart selection

When you perform various code refactorings that involve selecting an expression PhpStorm can help you select the expression of interest. This feature is known as smart expression selection .

The chapter for the Extract Variable refactoring is just an example. The smart expression selection feature, in fact, is available in all the refactorings that start with selecting an expression.

To use smart expression selection, follow these steps:

1. Place the cursor before or within the expression.

```
numbers.js
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
```

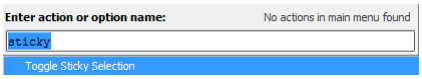
2. Choose Refactor | Extract Variable on the main menu or on the context menu. or press `Ctrl+Alt+V`.
3. In the Expressions pop-up menu, select the expression. To do that, click the required expression. Alternatively, use the Up and Down arrow keys to navigate to the expression of interest, and then press `Enter` to select it.

```
numbers.js
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("
  }
Expressions
"The number is "
"The number is " + i
document.write("The number is " + i)
```

Sticky selection

To toggle sticky selection, do one of the following:

- Press `Ctrl+Shift+A`, in the pop-up frame type sticky, and choose Toggle Sticky Selection from the suggestion list:



- In the Emacs keymap, use keyboard shortcut `Ctrl+Space`.

Tips and tricks

- When sticky selection is on, you can turn it off by invoking copy or cut, or by toggling it again.
- To create a large rectangular selection, create a normal selection first, with the given starting and ending points, and then press `Shift+Alt+Insert` to toggle to the column selection mode.

Cutting, Copying and Pasting

On this page:

- Basics
- Copying a selected fragment of text
- Copying the path to a file
- Copying the reference to a line or a symbol
- Cutting a selected fragment of text
- Pasting the last entry from the clipboard
- Pasting a specific entry from the clipboard

Basics

PhpStorm provides a number of handy Clipboard operations. You can copy, cut, and paste selected text, a path to a file, or a reference to a symbol or a line of code.

Because PhpStorm uses the system Clipboard, you can copy and paste between applications. So doing, when pasting Clipboard entries, PhpStorm removes any formatting from the text and any special symbols from the `String` values.

The Paste command smartly understands what is being inserted. If you paste a reference to a symbol, it is analyzed for possible imports, references, etc. So doing, PhpStorm provides the necessary brackets and places the caret at the appropriate insertion point. The Paste Simple command helps paste any Clipboard entry as a plain text, without any analysis.

PhpStorm enables Clipboard stacking, which means that you can store multiple Clipboard entries and access them with a single shortcut. The number of entries that can be kept in the Clipboard stack is customizable on the Editor page of the Settings/Preferences dialog.

Copying a selected fragment of text

Do one of the following:

- On the main menu, choose Edit | Copy.
- Press `Ctrl+C`.
- Click the Copy button on the toolbar.

Note that the `Ctrl+D` keyboard shortcut clones a line at the caret or a selected arbitrary fragment of text.

Copying the path to a file

Do one of the following:

- Open the desired file in the editor, then choose Edit | Copy Path on the main menu or press `Ctrl+Shift+C`.
- Select the desired file in the Project tool window and choose Copy Path on the context menu of the selection.

Copying the reference to a line or a symbol

1. Open the desired file in the editor.
2. Place the caret at a certain line of code.
3. Do one of the following:
   - On the main menu, choose Edit | Copy Reference
   - On the context menu of the line at caret, choose Copy Reference
   - Press `Ctrl+Shift+Alt+C`.

   PhpStorm creates a string in the format that depends on a symbol at caret. For example:

   `\MyNamespace\Catcher` for a PHP class

   `\MyNamespace\Catcher::$name` for a PHP variable

   `\MyNamespace\Catcher::set_experience` for a PHP method

Cutting a selected fragment of text

1. Select the desired fragment in the editor.
2. Do one of the following:
   - On the main menu, choose Edit | Cut.
   - Press `Ctrl+X`.
   - Click the Cut button on the toolbar.

Pasting the last entry from the clipboard

In the PHP context, you can paste fragments of code in two modes:

- In the Paste Simple mode, the fragment is inserted without escaping single quotes (`'`), double quotes (`"`), backslashes (`\`), and `<a href></a>`.
- In the Paste mode, PhpStorm escapes single quotes (`'`), double quotes (`"`), backslashes (`\`), and `<a href></a>` in the inserted fragment.

For example, if you insert `<a href="http://www.google.com/">go google</a>` into `<?php public $a = "|"; ?>`, where `|` indicates the cursor position, the result differs depending on the paste mode.

Paste Simple mode	Paste mode

```
<?php
public $a = "<a href="http://www.google.com/">go google</a>";
?>
```

```
<?php
public $a = "<a href=\"http://www.google.com/\">go google</a>";
?>
```

- To apply the Paste Simple mode, place the caret in the location where you want to paste the content and choose Edit | Paste Simple on the main menu or press `Ctrl+Shift+Alt+V`.
- To apply the Paste mode, place the caret in the location where you want to paste the content and choose Edit | Paste on the main menu, or press `Ctrl+V`, or click the Paste button on the toolbar.

Pasting a specific entry from the clipboard

1. On the main menu, choose Edit | Paste from History or press `Ctrl+Shift+V`.
2. In the Choose Content to Paste dialog box select the desired entry from the list of recent Clipboard entries, and click OK.

The depth of the Clipboard stack is configured in the Limits section on the Editor page of the Settings dialog box. When the specified number is exceeded, the oldest entry is removed from the list.

Copy and Paste Between PhpStorm and Explorer/Finder

PhpStorm enables tight interaction with the native file managers (Explorer on Windows, or Finder on Mac), and allows you exchange files and directories via the system clipboard, using numerous techniques:

- Cut, copy and paste files and directories from the Project tool window of PhpStorm to a directory in the file manager, and vice versa, using menu commands and keyboard shortcuts:

Keyboard shortcut	Function	Use this shortcut to...
Ctrl+C	Copy	Copy selected text to the Clipboard.

- Move (drag) or copy (Ctrl +drag) a file or directory from the file manager to a directory in the Project tool window.
- Move (drag) or copy (Ctrl +drag) a file from the Project tool window to a directory in the file manager.
- Open any file for editing, by dragging it from a file manager to the editor.

Using Drag-and-Drop in the Editor

PhpStorm allows copying and moving code fragments within the active editor tab, by means of drag-and-drop.

To move or copy code fragment

1. Make sure that using drag-and-drop is enabled in the Editor page of the IDE Settings.
2. Select the desired fragment in the editor.
3. Move or copy selection:
   - Move: Drag the selected fragment to the target location.
   - Copy: Keeping the `Ctrl` key pressed, drag selection to the target location.

Commenting and Uncommenting Blocks of Code

On this page:

- Basics
- Commenting and uncommenting lines of code
- Commenting and uncommenting blocks of code

Basics

You can comment or uncomment the current line or selected block of source code.

Commenting feature extends to all supported file types. For the custom file types, you can define line and block comments characters, as described in the section Creating and Registering File Types .

Commenting and uncommenting lines of code

Do one of the following:

- On the main menu, choose Code | Comment with Line Comment.
- Press `Ctrl+Slash`.

Commenting and uncommenting blocks of code

To add or remove a block comment, do one of the following:

- On the main menu, choose Code | Comment with Block Comment.
- Press `Ctrl+Shift+Slash`.

Saving and Reverting Changes

When working with PhpStorm, you don't need to worry about saving changed files: all changes are auto saved.

Unwanted changes can be undone at any stage of your development workflow. Any file or directory can be reverted to any of the previous states.

- When does PhpStorm auto save changed files?
- Tuning the autosave behavior
- Using the Save All command
- Marking files with unsaved changes in the editor
- Saving a file under a different name
- Reverting changes

When does PhpStorm auto save changed files?

Autosave is initiated by:

- Starting a run/debug configuration
- Performing a version control operation such as pull, commit, push, etc.
- Closing a file in the editor
- Closing a project
- Quitting the IDE

In fact, there is a lot more autosave triggers, and only the most important ones are mentioned above.

Tuning the autosave behavior

The following options are available for tuning the autosave behavior (File | Settings | Appearance and Behavior | System Settings):

- Save files on frame deactivation (i.e. on switching from PhpStorm to a different application)
- Save files automatically if application is idle for N seconds

Note that those are optional autosave triggers, and you cannot turn off autosave completely.

Using the Save All command

If necessary, you can initiate saving all changed files yourself. There is the Save All command for that:

- File | Save All
- `Ctrl+S`

Marking files with unsaved changes in the editor

Changed but yet unsaved files can be marked. For this purpose, there is the Mark modified tabs with asterisk option (File | Settings | Editor | General | Editor Tabs).

When this option is on, the files with unsaved changes have an asterisk * on their editor tabs.

Saving a file under a different name

There is no File | Save As command in PhpStorm. To save a file under a different name or in a different directory, use Refactor | Copy or `F5`.

Reverting changes

You can undo changes by using Edit | Undo or `Ctrl+Z`. To revert files to their previous states, use Local History and corresponding version control functionality .

Undoing and Redoing Changes

On this page:

- Basics
- How it works?
- Undoing and redoing changes

Basics

The Undo command discards the last changes to the file in the editor. The Redo command discards the results of the last Undo command.

You can undo or redo your changes as many times as required. However, when you exit PhpStorm, the undo history is lost.

PhpStorm smartly defines the logical steps that can be undone and redone. The following events signal about the end of a logical step:

- Pressing `Enter`.
- Repositioning the mouse cursor.
- Using navigation keyboard shortcuts.
- Cutting or pasting.
- Pressing `Tab`.

PhpStorm expands the undo and redo mechanism to complex operations, such as reformatting or refactoring source code, creating or deleting files. When you undo or redo a complex operation, PhpStorm requests for your confirmation.

How it works?

PhpStorm moves the caret before each step of undo/redo, and then performs the Undo/Redo actions.

Undoing and redoing changes

To undo an action, do one of the following

- On the main menu, choose Edit | Undo.
- Press `Ctrl+Z`.

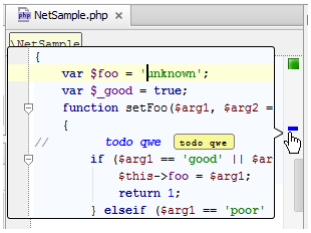
To redo an action, do one of the following

- On the main menu, choose Edit | Redo.
- Press `Ctrl+Shift+Z`.

'Lens' Mode

Hover the mouse pointer over a warning, error stripe or just some section on the scroll bar outside of the scroll box. PhpStorm shows the source code fragment annotated with the warning/error messages.

This helps viewing the context a marker applies to, and the source code outside of the editor visible area.



This behavior is enabled by default.

To toggle the lens mode

1. Do one of the following:
   - Open the Appearance page of the Settings dialog.
   - Right-click the code analysis marker on top of the current editor.

2. Select or clear the check box Show editor preview tooltip.

Multicursor

On this page:

– Basics

– Adding, deleting, and cloning carets

– Copying and pasting

Basics

PhpStorm supports multiple carets. The majority of the editor actions, such as keyboard navigation, text insertion and deletion, etc., apply to each caret. Live templates and autocompletion are supported as well.

It is possible to add or delete carets; at least one caret always exists in an editor tab.

The most recently added caret is considered primary. Highlighting of an editor line at caret applies to the primary caret only.

Placement and behavior of the carets depend on:

– Enabled or disabled column selection mode .

– Enabled or disabled placement of caret after the end of line .

Refer to the section Selecting Text in the Editor for additional information.

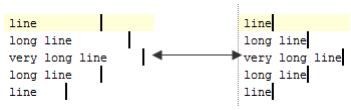
Adding, deleting, and cloning carets

To add carets, do one of the following

– Press `Alt` (Windows or UNIX)/ `Cmd` (OS X) and click the left mouse button at the location of the caret.

– Double-click `Ctrl` (Windows or UNIX)/ `Alt` (OS X) and press up or down arrow keys.

The new carets are added to the specified locations, according to setting of the Allow placement of caret after end of line check box:



To delete carets, do one of the following

– Press `Esc` to delete all the existing carets, except the primary one.

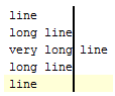
– Press `Shift+Alt` and click the left mouse button on a caret to be deleted.

To clone an existing caret upward or downward, do one of the following:

– Press `Ctrl+Shift+A` , type Clone caret, and choose the desired action from the suggestion list:

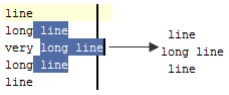
Note that by default these actions are not bound to the keyboard shortcuts. You can do it yourself, as described in the section Configuring keyboard shortcuts .

The primary caret is propagated upwards or downwards:



Copying and pasting

When a text with multiple cursors is copied (`Ctrl+C`) or cut (`Ctrl+X`), selections for each caret are placed to the clipboard. On paste (`Ctrl+V`), text from the clipboard is split into lines.



Scratches

On this page:

- Basics
- Creating scratch files
- Creating scratch buffers
- Observing the available scratches
- Closing scratches
- Deleting scratches
- Changing the language of a scratch
- Renaming, copying and moving scratches
- Important notes about scratches

Basics

PhpStorm provides a temporary editor. You can create a text or a piece of code for search or exploration purposes. PhpStorm suggests two types of temporary files:

Scratch files

The scratch files are fully functional, runnable, debuggable, etc. They require a language to be specified and have an extension. The scratch files are created via `Ctrl+Shift+Alt+Insert`.

Scratch buffers

The scratch buffers are only intended for pure editing, and as such they do not require specifying a language and do not have an extension. The scratch buffers belong to `.txt` type by default.

This action has no dedicated shortcut, but you can configure one as described in the section Configuring Keyboard Shortcuts .

Buffer files are reused after creating 5 files. So doing, after reuse, the content and language are reset.

Creating scratch files

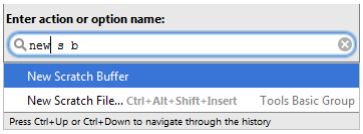
To create a scratch file

1. Do one of the following:
   - On the main menu, choose Tools | New Scratch File .
   - Press `Ctrl+Shift+Alt+Insert`
2. Select the language of the future scratch from the list that PhpStorm suggests. PhpStorm creates a temporary editor tab with the name `scratch.<extension>`. In the future, the default names will be `scratch_<number>.<extension>`.
3. Type the desired code.

Creating scratch buffers

To create a scratch buffer, follow these steps:

1. Press `Ctrl+Shift+A` or search everywhere .
2. Start typing the command name `New Scratch Buffer`:



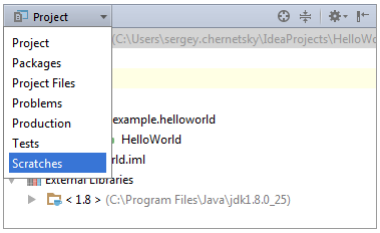
   PhpStorm creates a temporary editor tab with the name `buffer1`. In the future, the default names will be `buffer<number>`.
3. Type the desired code.

Note that though this action has no keyboard shortcut, you can still configure one as described in the section Configuring Keyboard Shortcuts . You can also switch from scratch files to scratch buffers by reassigning the shortcut to avoid garbage buildup.

Observing the available scratches

To observe the available scratch files and buffers, do one of the following:

- Choose Scratches view in the Project tool window :



- Press `Alt+F1` and choose Scratches (Navigating Between IDE Components).

Closing scratches

To close a scratch file or buffer, just click × on the editor tab. Refer to the section Closing Files in the Editor for details.

Deleting scratches

To delete a scratch file or buffer, follow these steps:

1. Switch to the Scratches view of the Project tool window .
2. Under the `Scratches` pseudo-folder, right-click the scratch to be deleted, and choose Delete on the context menu.
3. Confirm deletion.

Changing the language of a scratch

If you want to change the scratch's language when a scratch is already created, you can do so with the aid of the editor's context menu. This is how it's done:

1. Switch to the Scratches view of the Project tool window , and open for editing the scratch file or buffer you want to change language for.
2. Right-click the editor background, and choose Change Language (<current language>) on the context menu.
3. Select the desired language.

   Note the following:

   - Four latest items appear on top of the list before a separator.
   - You can narrow down the list by typing the language name.
   - You can assign a shortcut to this action as described in the section Configuring keyboard shortcuts .
   - Change Language action keeps extension in sync, if it exists.

Renaming, copying and moving scratches

PhpStorm makes it possible to perform rename refactoring of the scratches. To rename a scratch, follow these steps:

1. In the Project tool window , switch to the Scratches view and select the scratch to be renamed.
2. Press `Shift+F6`.

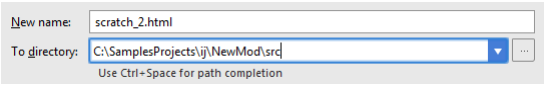
You can also perform renaming in the other ways:

- In NavBar : Jump to NavBar (`Alt+Home`) -> Rename (`Shift+F6`).
- In the Project tool window | Scratches view : Select In | Project | Scratches (`Alt+F1`) -> Rename (`Shift+F6`).
- Right from the editor: Refactor | Rename File.

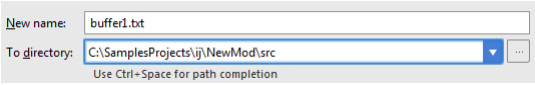
Copy and Move file actions are available the same way.

Note that when copying a scratch, PhpStorm includes the respective extension that corresponds to the file type. This is how it's done:

1. In the Project tool window , switch to the Scratches view and select the scratch to be copied.
2. Press `F5`. PhpStorm shows the following dialog box:



   This dialog box shows the scratch name with the corresponding extension. Note that when you copy a scratch buffer , the extension is `.txt`:



Important notes about scratches

Note the following:

- The scratch code in scripting languages is executable: you can run and debug it.
- Local history for scratches is supported.
- It is possible to perform clipboard operations with scratches.
- The scratches are stored, depending on your operating system,
  - Under PhpStorm home, in the directory `config/scratches` (on Windows/*NIX)
  - `~ Library->Preferences-><PhpStorm>XX->scratches` (on OS X)
- You can undo or redo changes in scratches.

Zooming in the Editor

PhpStorm makes it possible to change font size (zoom) in the active editor, and reset font size to the default value. These operations apply to the active editor only. In the other editor tabs, font size is not affected.

On this page:

– Enabling zooming

– Changing font size with the mouse wheel

– Restoring default font size

To enable changing font size in the editor

1. Open the Settings/Preferences dialog, expand the Editor node, and click General .

2. Make sure that the setting Change font size (Zoom) with Ctrl+MouseWheel is enabled.

To change font size using the mouse wheel

1. Place the caret in the editor.

2. While keeping the `Ctrl` / `⌘` key pressed, rotate the mouse wheel. As you rotate the mouse wheel forward, font size grows larger; as you rotate the mouse wheel backwards, font size decreases.
   The OS X users can use trackpad "Pinch-to-Zoom" gesture to change size of the font and the whole editing area.

To reset font size

1. Press `Ctrl+Shift+A`.

2. In the popup frame, type Reset font size, and click `Enter`.
   The default font size is restored.

Tip There is no default keyboard shortcut associated with Reset font size action. However, you can create your own shortcut as described in the section Configuring Keyboard Shortcuts .

Advanced Editing Procedures

This part describes more sophisticated editing techniques provided by PhpStorm:

- Adding, Deleting and Moving Code Elements
- Adding Editors to Favorites
- Changing Indentation
- Code Folding
- Viewing Current Caret Location
- Reformatting Source Code
- Toggling Writable Status
- Toggling Case
- Highlighting Braces
- Spellchecking
- Joining Lines and Literals
- Splitting Lines with String Literals
- Editing CSV and Similar Files in Table Format
- Using Macros in the Editor

Adding, Deleting and Moving Code Elements

On this page:

- Adding lines
- Duplicating lines
- Deleting lines
- Moving lines
- Moving statements
- Moving code element left or right

To add a line

- Press `Shift+Enter` to add a new line after the one where the caret is currently located and move the caret to the beginning of this new line.

  For instance, you have typed some text:

```
function multiplication (a,b){
    c = sum (a,b);
    return d;
}
```

  Press `Shift+Enter` to start the next line immediately:

```
function multiplication (a,b){
    c = sum (a,b);

    return d;
}
```

- To add a line before the current one, press `Ctrl+Alt+Enter`.

Tip Make sure that keyboard shortcuts are not in conflict. You can do that in the Keymap page of the Settings/Preferences dialog .

To duplicate a line or fragment

1. Place the caret at the line to be duplicated, or select the desired fragment of text.
2. Press `Ctrl+D`.

To remove a line

- Press `Ctrl+Y` to delete the line at caret.

To move a line

1. Place the caret at the line to be moved.
2. Do one of the following:
   - On the main menu, choose Code | Move Line Up or Code | Move Line Down.
   - Press `Shift+Alt+Up` or `Shift+Alt+Down`.

   PhpStorm moves the selected line one line up or down, performing the syntax check. For example:

```
var s = "The number is ";
document.write("<br/>");
document.write(s + i);
```

   After moving line at caret:

```
var s = "The number is ";
document.write(s + i);
document.write("<br/>");
```

To move a statement up or down

1. Select a statement to be moved, or just place the caret at the first or last lines of a multi-line statement. Note that if moving a statement is not allowed in the current context, the commands will be disabled.
2. Do one of the following:
   - On the main menu, choose Code | Move Statement Up/Move Statement Down.
   - Press `Ctrl+Shift+Up` or `Ctrl+Shift+Down`.

   If you apply the same commands to a line at caret, or a to a selection, it will be moved one line up or down.

   PhpStorm moves the selected statement above the previous one, or directly underneath the next one, performing the syntax check. For example, place the caret at the method declaration:

```
    for ($i=0;$i<=10;$i++) {
        if (($i % 2) == 0) {
            echo "The number is even." . "<br>";
        }
        else {
            echo "The number is odd." . "<br>";
        }
        echo "The number is " . $i . ".";
    }
```

After moving the statement:

```
    for ($i=0;$i<=10;$i++) {
        echo "The number is " . $i . ".";
        if( ( $i % 2 ) == 0 ) {
            echo "The number is even." . "<br>";
        } else {
            echo "The number is odd." . "<br>";
        }
    }
```

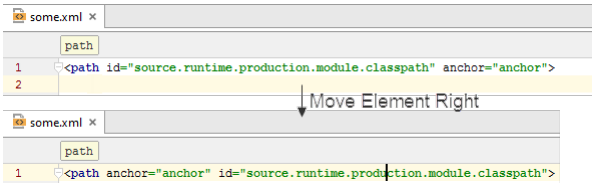
To move code element to the left or to the right

1. Place the caret at the desired code element, or select the elements to be moved.

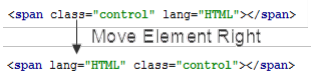
2. Do one of the following:
   - On the main menu, choose the commands Code | Move Element Left or Code | Move Element Right
   - Press `Ctrl+Shift+Alt+Left` or `Ctrl+Shift+Alt+Right`

Examples of code elements for which this functionality is currently implemented:

- XML: tag attributes.



- HTML: tag attributes.



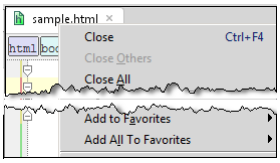
Adding Editors to Favorites

You can group most needed items into Favorite lists and get quick access to them through the Favorites tool window.

To add one or more items to Favorites

1. Do one of the following:
   - Open the desired files in the Editor.
   - Select one or more items in the Project tool window.

2. Right-click the editor tab or the selection in the Project tool window, and choose Add to Favorites on the context menu.



3. On the submenu, specify the Favorites list to add the selected items to. Do one of the following:
   - To add the items to an existing list, select the desired list in the submenu.
   - To create a new list, choose Add to New Favorites List. In the Add New Favorites List dialog box that opens enter the desired group name or accept default settings.

Changing Indentation

PhpStorm makes it possible to:

- Indent or unindent text. This action applies to a selection, or to a line at caret.
- Fix wrong indentation according to the code style.
- Choose tabs or spaces for indentation. This action applies to a selection, or the whole current file in the active editor.

To change indentation of a text fragment, do one of the following

- On the main menu, choose Edit | Indent Selection / Edit | Unindent Selection.
- Press `Tab` / `Shift+Tab`.

To fix indentation

Sometimes it is necessary to change indentation of a line at caret.

1. Place the caret at a line with wrong indentation.
2. Press `Ctrl+Alt+I`.

To toggle between tabs and spaces

- On the main menu, choose Edit | Convert Indents , and then choose To Spaces or To Tabs respectively.

Code Folding

On this page:

- Basics
- Code folding means
- Folding predefined and custom regions
- Commands of the Folding menu and associated shortcuts
- Specifying code folding preferences
- Viewing folded code fragments
  - Viewing the beginning of a folding region
- Using code folding comments
  - Supported folding comments
  - Surrounding a fragment with folding comments
  - Navigating to folding regions

Basics

You can collapse (fold) code fragments reducing them to a single visible line. In this way, you can hide the details that, at the moment, seem unimportant. If and when necessary, the folded code fragments can be expanded (unfolded).

Folded code fragments, normally, are shown as shaded ellipses (`...`).

Code folding means

You can collapse and expand code fragments by using:

- Code folding toggles (, or). These toggles are shown in the editor to the left of the corresponding folding regions. If a region is unfolded, indicates the beginning of the region while is located at its end. For folded regions, the toggle is shown as .
  If you hold the `Alt` key and click , or , the region is collapsed or expanded recursively, i.e. along with all its subordinate regions.
- Commands of the Folding menu and associated keyboard shortcuts. The Folding menu can be accessed from the main menu bar (Code | Folding), or as a context menu in the editor. The shortcuts are shown right in the menu. See Commands of the Folding menu and associated shortcuts .
- Folded fragments themselves: click `...` to expand the corresponding fragment. See also, Viewing folded code fragments .

Folding predefined and custom regions

You can fold and unfold:

- Predefined regions that correspond to such elements as import declarations, method bodies, documentation comments, etc. The predefined regions, roughly, correspond to the ones listed under Collapse by default on the Editor | General | Code Folding page in the Settings/Preferences dialog.
  For the predefined regions, the folding toggles are available right away, without the need to perform any additional actions.
- Any selected code fragment. A custom folding region for a selection is created and removed by means of the Fold Selection/ Remove Region command (`Ctrl+Period`).
- Regions surrounded by corresponding commented folding markers (e.g. `//<editor-fold desc="Description">...//</editor-fold>`). See Using code folding comments .

Note Code folding works for the keywords `if` / `while` / `else` / `for` / `try` / `except` / `finally` / `with` in case of at least two statements.

Commands of the Folding menu and associated shortcuts

The Folding menu can be accessed from the main menu bar (Code | Folding), or as a context menu in the editor.

Command	Shortcut	Description
Expand	`Ctrl+NumPad Plus`	Expand the current collapsed fragment
Collapse	`Ctrl+NumPad -`	Collapse the current folding region
Expand Recursively	`Ctrl+Alt+NumPad Plus`	Expand the current folded fragment and all the subordinate collapsed folding regions within that fragment
Collapse Recursively	`Ctrl+Alt+NumPad -`	Collapse the current folding region and all the subordinate folding regions within it
Expand All	`Ctrl+Shift+NumPad Plus`	Expand all collapsed fragments within the selection, or, if nothing is selected, expand all the collapsed fragments in the current file
Collapse All	`Ctrl+Shift+NumPad -`	Collapse all folding regions within the selection, or, if nothing is selected, collapse all the folding regions in the current file

Expand to level | 1, 2, 3, 4 or 5	`Ctrl+NumPad *, 1`
	`Ctrl+NumPad *, 2`
	`Ctrl+NumPad *, 3`
	`Ctrl+NumPad *, 4`
	`Ctrl+NumPad *, 5`
		Expand the current fragment and all the nested fragments up to the specified level

Expand all to level | 1, 2, 3, 4 or 5	`Ctrl+Shift+NumPad *, 1`
	`Ctrl+Shift+NumPad *, 2`
	`Ctrl+Shift+NumPad *, 3`
	`Ctrl+Shift+NumPad *, 4`
	`Ctrl+Shift+NumPad *, 5`
		Expand all the collapsed fragments in the file up to the specified nesting level

Expand doc comments		Expand all documentation comments in the current file
Collapse doc comments		Collapse all documentation comments in the current file
Fold Selection / Remove region	Ctrl+Period	Collapse the selected fragment and create a custom folding region for it to make it "foldable" / Expand the current fragment and remove the corresponding custom folding region to make the fragment "unfoldable"

Specifying code folding preferences

You can specify:

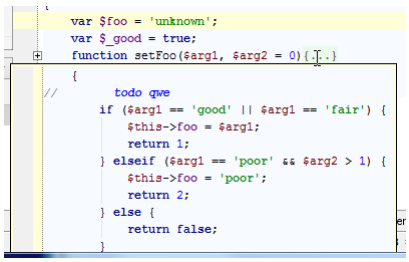
- Whether the code folding toggles should be shown.
- Which folding regions should be collapsed by default.

The corresponding settings are in the Settings dialog (Ctrl+Alt+S) on the Editor | General | Code Folding page.

For more information, see Code Folding page .

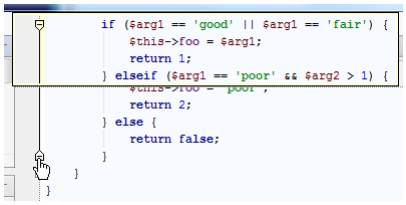
Viewing folded code fragments

To see the contents of a folded fragment, point to the ellipsis ... that indicates that fragment.



Viewing the beginning of a folding region

To see the beginning of a folding region - if it's not currently visible - point to the folding toggle at the end of that region.



Using code folding comments

- Supported folding comments
- Surrounding a fragment with folding comments
- Navigating to folding regions

Supported folding comments

You can create custom folding regions by surrounding code fragments with the commented lines. So doing, the comments can be either NetBeans style, or Visual Studio style.

NetBeans style

```
//<editor-fold desc="Description">
                    ...
                    //</editor-fold>
```

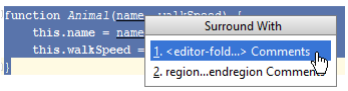
Visual Studio style

```
//region Description
                    ...
                    //endregion
```

Once you have chosen the style for a file, don't use the other style in that file.

Surrounding a fragment with folding comments

1. Select the code fragment of interest.
2. Select Code | Surround With or press Ctrl+Alt+T .
3. Select the folding comments to be used.



4. Specify the fragment description.

```
//<editor-fold desc="MyRegion">
function Animal(name, walkSpeed) {
    this.name = name
    this.walkSpeed = walkSpeed
}
//</editor-fold>
```

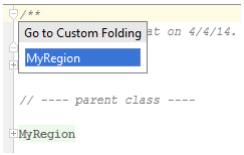
Now if you collapse the fragment, the description you have specified is shown in place of the code.

```
MyRegion
```

Navigating to folding regions

You can navigate to custom folding regions that were formed by surrounding code fragments with the corresponding commented folding markers:

1. Select Navigate | Custom Folding or press `Ctrl+Alt+Period`.
2. Select the target folding region. (The regions in the list are identified by their descriptions.)



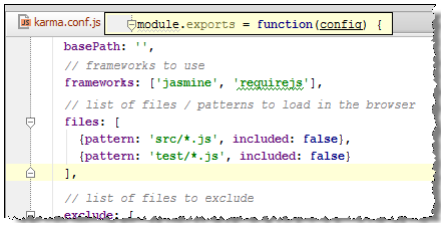
Viewing Current Caret Location

If in course of editing, searching, or navigating through a file, the cursor position runs out of the visible editor area, above the upper editor edge, you don't need to scroll through the file to obtain instant information about the current caret location.

To view the current caret position, do one of the following

- On the main menu, choose View | Context Info
- Press `Alt+Q`.

The pop-up frame appears on top of the editor, showing the name of the class, method, or function where the caret currently resides:



Reformatting Source Code

On this page:

- Basics
- Reformatting code
- Reformatting code for the current file
- Skipping a region when reformatting source code
- Example of using formatting markers

Basics

PhpStorm lets you reformat source code to meet the requirements of your code style. PhpStorm will lay out spacing, indents, keywords etc. Reformatting can apply to the selected text, entire file, or entire project.

It is also possible to apply reformatting to the parts of the source code only, using the formatting markers .

Reformatting code

To reformat code for a directory, follow these steps:

1. In the Project tool window, select a directory you want to apply your reformatting to.
2. From the main menu, select Code | Reformat Code or press `Ctrl+Alt+L`.
   Alternatively, in the Project tool window , right-click the directory and from the context menu, select Reformat Code.
3. In the Reformat Code dialog box, specify the necessary options and filters for you reformatting and click Run.

Reformatting code for the current file

To reformat code for the current file, follow these steps:

1. In the editor of the currently opened file, press `Ctrl+Shift+Alt+L`.
   Note that if you select Code | Reformat Code from the main menu or press `Ctrl+Alt+L`, PhpStorm will try to reformat the source code automatically without opening the Reformat File dialog.
2. In the Reformat File dialog, specify options for the reformatting and click Run.

Skipping a region when reformatting source code

To enable formatter markers, make sure to select the check box Enable formatter markers in comments in the Code Style page of the Settings/Preferences dialog, and type the markers in the Formatter off/on fields.

To skip a certain region on reformatting, follow these steps:

1. At the beginning of the region, create a line comment (`Ctrl+Slash`), and then manually type the marker specified in the Formatter off field of Code Style page.
2. At the end of the region, create a line comment (`Ctrl+Slash`), and then manually type the marker specified in the Formatter on field of Code Style page.
3. Perform code reformatting, as described above.

Alternatively, create a live template to surround a block of code with formatter off/on markers, see Creating and Editing Live Templates .

Example of using formatting markers

The original source code

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested))
    {
        return false;
    }
//@formatter:on
require_once $paths['public'] . '/index.php';
```

The code after reformatting

When the formatting markers are disabled, the original formatting is broken:

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested)) {
    return false;
}
//@formatter:on
require_once $paths['public'] . '/index.php';
```

When the formatting markers are enabled, the original formatting is preserved:

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested))
    {
        return false;
    }
//@formatter:on
require_once $paths['public'] . '/index.php';
```

Toggling Writable Status

If a file is read-only, it is marked with the closed lock icon in the Status bar , in its editor tab, or in the Project tool window. If a file is writable, then it is marked with the open lock icon in the Status bar. This section describes how to change read-only attribute of a file.

Tip If a read-only status is set by a version control system, it's suggested that you use PhpStorm version control integration features. For more information, see Concepts of Version Control .

To toggle read-only attribute of a file

1. Open file in the editor, or select it in the Project tool window .

2. Do one of the following:
   - On the main menu, choose File | Make File Read-Only, or File | Make File Writable.
   - Click the lock icon in the Status bar .

Toggling Case

On this page:

- Toggling between upper and lower cases
- Tips and tricks

Toggling between upper and lower cases

To toggle between upper case and lower case

1. Select text fragment, or just place the caret at the line you want to change case in.
2. On the main menu, choose Edit | Toggle Case, or press `Ctrl+Shift+U`.

Tips and tricks

Did you know that

- Applying Toggle Case to CamelCase name format converts the name format to lower case?
- The lowercase names are converted to the uppercase format?
- Applying Toggle Case to the uppercase name format converts the name format to lower case?

Highlighting Braces

This editor feature significantly improves readability of the code, and simplifies search for unclosed blocks or tags.

To highlight block borders

- Place the caret immediately after the block closing brace/bracket or before block opening brace/bracket.

If the editor can find the block border, its braces, brackets or tags are highlighted with blue and a blue outline appears in the gutter area.

If the opening brace, bracket or tag is currently out of sight, you don't need to scroll to the beginning of the block. The editor shows a pop-up window on top that displays the beginning of the block.

If the editor cannot find the pair brace, bracket or tag, the unmatched one is highlighted with pink when the caret is placed next to it, and is underlined with a red curly line.

Spellchecking

On this page:

- Basics
- Checking the spelling of a word
- Configuring the dictionaries to use
- Configuring spellchecking options

Basics

PhpStorm helps you make sure that all your source code, including textual strings, comments, and literals, and commit messages, is spelt correctly. For this purpose, PhpStorm suggests a dedicated Typo inspection, which is supported by the corresponding bundled plugin and is enabled by default.

Correctness of spelling is checked against pre-defined dictionaries (as of now, `jetbrains.dic` and `english.dic`), and any number of user-defined custom dictionaries.

A user dictionary is a textual file with the `dic` extension, containing the words you want to be accepted by the Typo inspection as correct. The words in such dictionaries are delimited with the newline.

Besides that, you can define your own list of words that will be skipped by the inspection. You can add words to this list "on-the-fly", or intentionally while setting up your spellchecker options.

With the Typo inspection enabled, PhpStorm detects and highlights words not included in dictionaries and user's words list. It up to the user to provide correct spelling, accept word as is, or disable inspection.

If a word is accepted, it will be added to the user's words list, and skipped by the spellchecker in future. If inspection is disabled, all typos will be ignored.
In the textual strings and comments, spelling of a word at caret can be changed to a correct one. In the contexts that enable Rename refactoring, the inspection suggests to rename all occurrences of a symbol.

Checking the spelling of a word

1. Place the caret on a word highlighted by the Typo inspection.
2. Press `Alt+Enter` to show the available intention actions.
3. Choose one of the following actions:
   - Change to: select the desired spelling of a textual string or comment from the suggestion list.
   - Save to dictionary: add word to the user's list and skip it in future.

Configuring the dictionaries to use

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Spelling under Editor.
2. Define the set of dictionaries to be used for spellchecking.
   1. Switch to the Dictionaries tab.
   2. The Dictionaries area in the bottom of the page shows a list of the dictionaries that can be used in spellchecking. The list contains the dictionaries that come bundled with PhpStorm by default and user-defined dictionaries detected in the folders from the Custom Dictionaries Folder area above.
      - To have a default dictionary applied in the current project, select the check box next to it.
      - To exclude a default dictionary from spellchecking within the scope of the current project, clear the check box next to it.
   3. In the Custom Dictionaries Folder area, configure your custom dictionaries to use. This area displays a list of directories that contain user-defined dictionary files (text files with the `dic` extension, containing words separated with a newline).
      - To add a new folder to the list, click + and choose the required folder in the Select Path Dialog dialog that opens .
        The full path to the folder is added to the Custom Dictionaries Folder list, and all the `*.dic` files found in this folder are added to the Dictionaries list.
      - To remove a folder from the list, select it and click —.
3. Besides configuring a custom dictionary, you can create your own Word List with the words that you want to be skipped during spell checking without being included in a custom dictionary.
   1. Switch to the Accepted Words tab.
   2. Create a Word List:
      - Click the + icon to open the Add New Word dialog box and specify a new entry there. CamelCase or snake_case are not supported. If you try to add a word that is already included in one of the spelling dictionaries, PhpStorm displays an error message The word <just typed word> is already in the dictionary.
      - To remove an item from the list, select it and click —.

Configuring spellchecking options

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Inspections under Editor.
2. On the Inspections page, that opens showing a list of inspection types, expand the Spelling node and click Typo in the central pane.
3. In the right-hand pane, configure the Typo inspection:
   - In the Options area, define the type of contents to be inspected by selecting or clearing the Process Code, Process Literals, and Process Comments check boxes.
   - In the Severity area, choose the inspection severity level and the scope to apply this level in.

Joining Lines and Literals

PhpStorm makes it possible to concatenate two unselected or several selected lines into one, removing the extra spaces, and providing the proper syntax. This operation smartly analyzes the lines being joined and treats them accordingly. For example, you can join lines of code, lines of comments, field declaration and initialization.

On this page:

– Joining lines

– Joining string literals

Joining lines

1. Place the caret on the line, to which the other lines should be added.
2. Sequentially press `Ctrl+Shift+J` keyboard shortcut, until all fragments are joined in a single line.

> **Tip** You can select the lines and press `Ctrl+Shift+J` to obtain the same result.

Joining string literals

1. Select the lines with string literals that should be joined.
2. Press `Ctrl+Shift+J` keyboard shortcut. All redundant characters (spaces, quotes, and plus signs) are gone.

Splitting Lines With String Literals

To split a string literal into two parts:

1. Set the caret in the string literal to be split.
2. Press `Enter`.

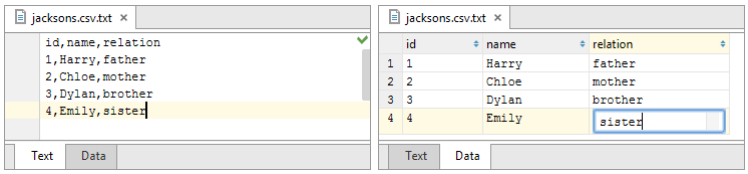
Editing CSV and Similar Files in Table Format

On this page:

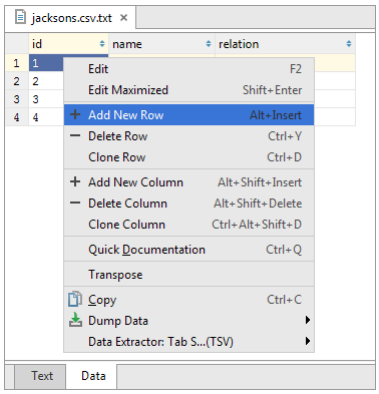
- Overview
- Prerequisites
- Opening the table editor
- Sorting data
- Hiding and showing columns
- Transposing the table
- Enabling coding assistance for a column
- Modifying cell contents
- Adding and deleting rows and columns
- Copying data to the clipboard or saving them in a file
- Changing data output format
- The Data tab is not visible. Why?

Overview

For text files containing delimiter-separated values (e.g. CSV, TSV), PhpStorm provides an alternative, table editor.



Most of the functions in the table editor are accessed as context menu commands. Many of the commands have keyboard shortcuts.



Note that the context menus for the header row and the rest of the table are different.

Prerequisites

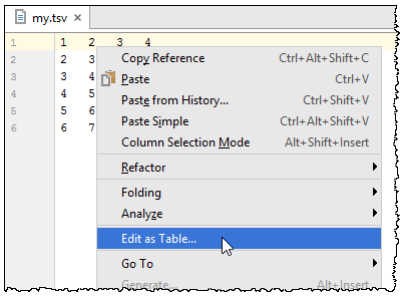
For the table editor and associated features to be available:

- The Database Tools and SQL plugin must be enabled. (This plugin is bundled with the IDE and enabled by default.)
- The file name extension must be associated with the text file type. See e.g. File Types .

Opening the table editor

You can open the table editor for a whole file or for its fragment.

1. Open the file of interest in the editor.
2. If you want to open the table editor for a fragment, select that fragment.
3. Select Edit as Table from the context menu.
4. In the dialog that opens , specify conversion setting and click OK. After that, the Text and Data tabs appear.



Sorting data

You can sort table data by any of the columns by clicking the cells in the header row.

Each cell in this row has a sorting marker in the right-hand part and, initially, a cell may look something like this: name . The sorting marker in this case indicates that the data is not sorted by this column.

If you click the cell once, the data is sorted by the corresponding column in the ascending order. This is indicated by the sorting marker appearance: name ▲1. The number to the right of the marker (1 on the picture) is the sorting level. (You can sort by more than one column. In such cases, different columns will have different sorting levels.)

When you click the cell for the second time, the data is sorted in the descending order. Here is how the sorting marker indicates this order: name ▼1.

Finally, when you click the cell for the third time, the initial state is resorted. That is, sorting by the corresponding column is canceled: name ⇕.

Hiding and showing columns

To hide a column, right-click the corresponding header cell and select Hide column.

To show a hidden column:

1. Do one of the following:
   - Right-click any of the cells in the header row and select Column List.
   - Press `Ctrl+F12`.

   In the list that appears, the names of hidden columns are shown struck through.
2. Select (highlight) the column name of interest and press `Space`.
3. Press `Enter` or `Escape` to close the list.

Transposing the table

The transposed table view is available. In this view, the rows and columns are interchanged.

To turn this view on or off, use the Transpose context menu command.

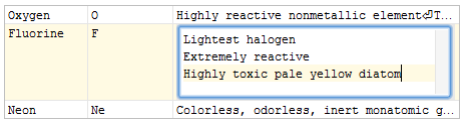
Enabling coding assistance for a column

You can assign a column one of the supported languages (e.g. SQL, HTML or XML): right-click the corresponding header cell, select Edit As and select the language. As a result, you get coding assistance for the selected language in all the cells of the corresponding column.

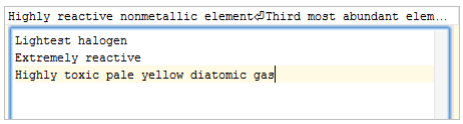
You can also assign a language to an individual cell .

Modifying cell contents

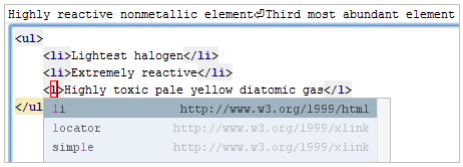
1. To start editing a value, do one of the following:
   - Double-click the corresponding table cell.
   - Right-click the cell and select Edit Value or Edit Maximized from the context menu.
   - Select the cell and press `F2` or `Shift+Enter`. In the latter case, the cell will be maximized.
   - Select the cell and start typing. Note that in this case the initial cell contents are deleted right away and is replaced with the typed value.
2. When in the editing mode, you can:
   - Modify the value right in the cell. To start a new line, use `Ctrl+Enter`. To save the changes, press `Enter`. To restore an initial value and quit the editing mode, press `Escape`.



   - Maximize the cell if you need more room for editing. To do that, press `Ctrl+Shift+Alt+M`, or right-click the cell and select Maximize.
     When working in a maximized cell, use `Enter` to start a new line and `Ctrl+Enter` to save the value. To restore an initial value and quit the editing mode, press `Escape`.



   - Insert the contents of a text file into the cell. To do that, right-click the cell and select Load File. Then select the necessary file in the dialog that opens.
   - Edit a value in the cell as a fragment in one of the supported languages (e.g. SQL, HTML or XML). To do that, right-click the cell, select Edit As and select the language. As a result, you get coding assistance for the language you have selected.



Adding and deleting rows and columns

Use the following context menu commands and shortcuts:

- Add New Row (`Alt+Insert`).
- Delete Row (`Ctrl+Y`). To delete more than one row at once, first, select the corresponding rows or cells in the corresponding rows.
- Clone Row (`Ctrl+D`). This command creates a copy of the current row.
- Add New Column (`Shift+Alt+Insert`).
- Delete Column (`Shift+Alt+Delete`). To delete more than one column at once, first, select the cells in the corresponding columns.
- Clone Column (`Ctrl+Shift+Alt+D`). This command creates a copy of the current column.

Copying data to the clipboard or saving them in a file

1. Use one of the following context menu commands:
   - Copy (`Ctrl+C`). This command copies the data for the selected cells to the clipboard.
     If the table is shown transposed (see Transposing the table), the range of cells is copied transposed (as shown) if the Allow Transposition option is on and non-transposed (as in the original table) otherwise.

     To turn the Allow Transposition option on or off, right-click the table, point to Data Extractor: <current_format> and click Allow Transposition.
   - Dump Data | To Clipboard. This command copies the data for the whole table to the clipboard.
   - Dump Data | To File. This command saves the data for the whole table in a file. Before actually saving the data, the dialog is shown which lets you select the output format and see how your data will look in a file.
2. If you are saving the data in a file, specify the file name and location.

See also, Changing data output format.

Changing data output format

When you copy table data to the clipboard (the Copy and Dump Data | To Clipboard commands), unless you change the format, the data are output as Tab-Separated Values (TSV).

To change the output format or to define a new format for the output:

1. Right-click the table and point to Data Extractor: <current_format>.
2. Do one of the following:
   - To select a different output format, click the name of the format.
   - To define a new delimiter-separated values format, or to view or modify the settings for an existing one, click Configure CSV Formats. As a result, the CSV Formats Dialog will open.

The Data tab is not visible. Why?

For the table view and associated features to be available:

- The Database Tools and SQL plugin must be enabled. (This plugin is bundled with the IDE and enabled by default.)
- There must not be empty lines in your file, e.g. at the end.
- The file name extension must be associated with the text file type.
- The file should contain at least two lines and two "columns".
- Data tab is invoked by the context menu command Edit as Table.

Using Macros in the Editor

Macros provide a convenient way to automate repetitive procedures you do frequently while writing code. You can record, edit and playback macros, assign them a shortcut, and share them. Generally speaking, macros are designed for rather simple operations, and as such have the following limitations:

- Macros can be used for editor-related actions within a file.
- You cannot record such actions as button clicks, navigating to pop-up dialog boxes, and accessing tool windows or menus.

If a macro is intended for temporary use only, it is unnamed; permanent macros have unique names.

This section describes how to:

- Record Macros
- Bind Macros With the Keyboard Shortcuts
- Play Back Macros
- Edit Macros

To bind a macro with a keyboard shortcut

1. Open the Settings dialog and click Keymap .
2. Create a new keymap Run/Debug Configuration: PHP Script or select an editable keymap from the list of keymaps.
3. Expand the Macros node and select the macro for which a keyboard shortcut should be created.
4. Right-click on the macro and choose Add Keyboard Shortcut in the context menu.
5. In the Enter Keyboard Shortcut dialog, press the keys to be used as a shortcut. The keystrokes are immediately reflected in the First Stroke field. Optionally, select the Second stroke check box and specify the second stroke. As you press the keys, the Preview field displays the keystrokes you pressed, and the Conflicts field displays warnings, if the keystrokes are already in use.
6. Click OK using the mouse pointer to create a shortcut and bind it with the macro.

   Tip It is important that you use the mouse pointer, because any keystroke is interpreted as a shortcut.

7. Click Apply to save the settings.

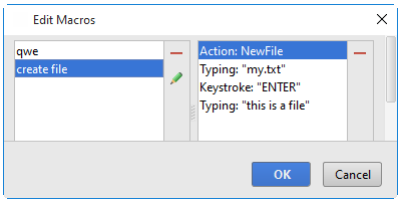
Editing Macros

After recording, you can remove or rename any or all of the macros from the list of available macros. The actions list of each macro is also editable - you can remove unnecessary actions.

To edit macros

1. On the main menu, choose Edit | Macros | Edit macros.

2. In left-hand pane of the Edit Macros Dialog , select the macro to be edited or deleted:



3. To delete a macro, click . To change the macro name, click , and specify the new name in the Rename Macro dialog.

4. To change the list of actions for a macro, select an action in the action list, and click .

Playing Back Macros

You can play back the recorded macros using the Edit | Macros menu commands, or custom shortcuts .

To play back a temporary macro

– On the main menu, choose Edit | Macros | Play Back Last Macro.

To play back a named macro

– On the main menu, choose Edit | Macros | <Macro name>.

To play back a macro with a keyboard shortcut

1. Select a keymap with the macro bindings on the Keymap settings page.
2. Press keyboard shortcut that corresponds to the desired macro.

To record a macro

1. On the main menu, choose Edit | Macros | Start Macro Recording. From that moment on, all your recordable actions are recorded.
2. When you are done with the procedure, choose Edit | Macros | Stop Macro Recording.
3. In the Enter Macro Name dialog, specify the name of the new macro, and click OK. If the macro is intended for temporary use only, you can leave the name blank.

Managing Editor Tabs

In this section:

- Managing Editor Tabs
  - Introduction
- Configuring Behavior of the Editor Tabs
- Navigating Between Editor Tabs
- Pinning and Unpinning Tabs
- Splitting and Unsplitting Editor Window
- Detaching Editor Tabs
- Editing Multiple Files Using Groups of Tabs
- Changing Placement of the Editor Tab Headers
- Sorting Editor Tabs

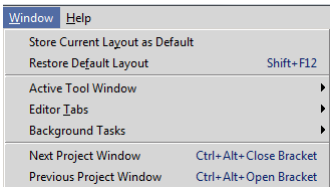
Introduction

Every time you open a file for editing, a dedicated tab is added to the editor window, next to the active editor tab.

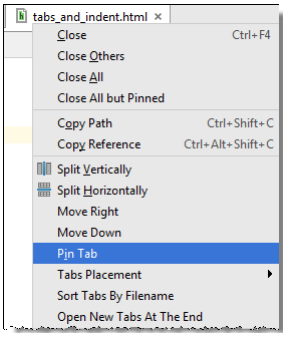
PhpStorm can limit the number of tabs opened in the editor simultaneously. When the number of tabs reaches its limit, the editor closes the tabs according to the tab closing policy, that is defined in the Editor Tabs settings page. By default, the tab limit is 10, but you can change it if necessary.

All commands, related to managing editor tabs, are available from:

- Window | Editor Tabs menu.



- Context menu of a tab.



Configuring Behavior of the Editor Tabs

On this page:

- Changing the number of editor tabs
- Disabling editor tabs
- Tips and tricks

Changing the number of editor tabs

To change the maximum allowed number of tabs

1. Open the Editor settings. To do that, click on the main toolbar, then click the Editor node in the Settings dialog box that opens.
2. In the Editor Tabs page of the Editor settings, type the desired maximum allowed number of the editor tabs to be opened at a time in the Tab limit field.

Disabling editor tabs

To disable editor tabs

- In the Editor Tabs page of the Editor settings, select None from the Tab placement drop-down list.

Tips and tricks

- If the tab limit equals to 1, the tabs will be disabled. If you want the editor to never close the tabs, type some unreachable number.
- With the disabled tabs, use the View | Recent files (`Ctrl+E`) command to quickly switch between files.

Navigating Between Editor Tabs

On this page:

- Navigating between editor tabs
- Navigating though the previously visited tabs
- Viewing all opened editor tabs and choosing the active editor

To navigate from the current tab to the next or previous tab

- Right-click the current editor tab and choose Select Next/Previous Tab on the context menu.
- Press `Alt+Right` or `Alt+Left`. So doing, the focus moves to the editor tab located next to the right or to the left from the active editor tab.
- Press `Ctrl+Tab` to use the Switcher .

This approach allows jumping from one tab to another as your editing session requires. While you move between the editor tabs, PhpStorm remembers the caret position within each opened file.

To go back and forth through the history of visited tabs

- On the main toolbar, click or .
- On the main menu, choose Navigate | Back / Forward.
- Press `Ctrl+Alt+Left` or `Ctrl+Alt+Right`.

This approach enables you to move back and forth through the history of your navigation, same way as it is done in a Web browser. As you move from file to file during your editing session, PhpStorm keeps track of the visited locations and enables you to go back, using the Navigate | Back / Forward commands.

Note On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.

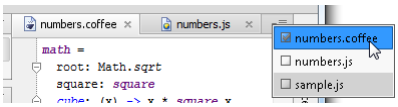
To view all editor tabs and select the active editor

If all opened tabs are shown in a single row, some of the tabs may become invisible. PhpStorm helps display the list of the opened editor tabs that did not fit in the editor window and have become invisible whereupon you can choose the tab to activate.

1. Do one of the following:
   - On the main menu, choose Window | Editor Tabs | Show Hidden Tabs
   - Click

   The list of all the tabs that are opened but invisible appears. So doing the names of the tabs, which are currently visible, are displayed on the light background; the names of the tabs outside of the main window are shown on the darker background.

2. Click the name of the desired editor tab:



   The selected editor tab becomes active and gets the focus.

The command Show All Tabs and the icon are only available when:

- The Show tabs in single row check box is selected in the Editor tabs page of the Editor settings.
- Some of the opened tabs are not visible because they do not fit in the editor window.

Pinning and Unpinning Tabs

On this page:

– Basics

– Pinning an editor tab

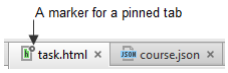
– Unpinning a tab

Basics

PhpStorm can limit the number of tabs opened in the editor simultaneously. When the number of tabs reaches its limit, the editor closes the tabs according to the tab closing policy, that is defined in the Editor Tabs settings page. By default, the tab limit is 10, but you can change it if necessary.

To prevent a tab from being closed automatically, you can pin this tab. Besides, when you close the editor tabs, you have an option to preserve pinned tabs opened and close only the unpinned tabs.

When a tab is pinned, there is a special marker on it.

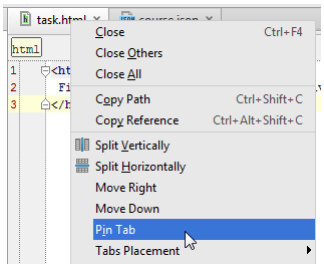


Tip The pinned status of a tab is helpful when you open a file for reference, rather than for editing.

Pinning an editor tab

1. Switch to the desired editor tab.

2. Right-click the editor tab, and choose Pin Tab on the context menu:



Unpinning a tab

1. Switch to the desired editor tab.

2. Right-click the editor tab, and choose Unpin Tab on the context menu.

Splitting and Unsplitting Editor Window

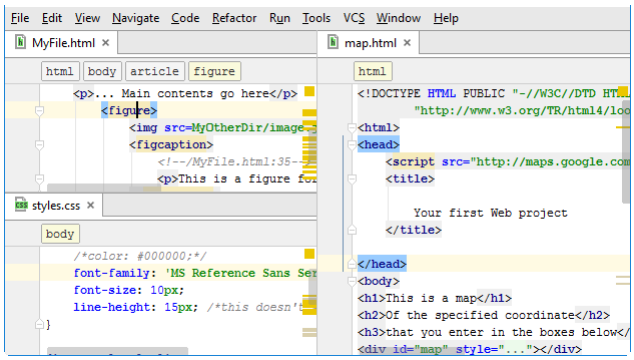
On this page:

- Basics
- Splitting editor tab
- Changing splitter orientation
- Removing a splitter

Basics

Splitting the editor window divides it into independent panes. You can split the editor window into as many panes as required, each one containing multiple tabs.

Each pane can be allocated vertically or horizontally. Thus, splitting helps create different editor layouts, organize tabs into groups, and edit multiple files simultaneously . For example, you can scroll through a part of a file, having at the same time the possibility to view the lines in its other part.



Splitting editor tab

To split an editor tab creating a file copy

1. Switch to the desired tab.
2. Right-click the tab header and choose Split Vertically or Split Horizontally on the context menu.

To split an editor tab without copying a file

1. Switch to the desired tab.
2. Right-click the tab header and choose Move Right or Move Down on the context menu.

Changing splitter orientation

To change splitter orientation

1. Switch to the desired tab.
2. Right-click the tab header and choose Change Splitter Orientation on the context menu.

Removing a splitter

To remove splitter

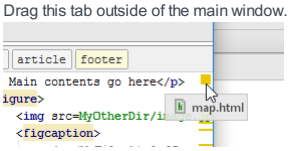
1. Switch to the desired tab.
2. Right-click the tab header and choose one of the following commands on the context menu:
   - To remove splitting in the active tab, choose Unsplit.
   - To remove splitting in all the open editor tabs, choose Unsplit All.

Detaching Editor Tabs

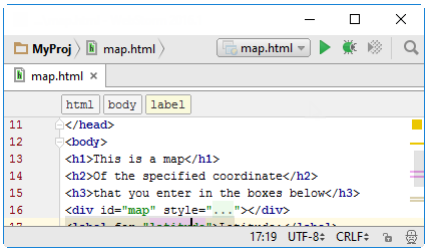
PhpStorm makes it possible to detach editor tabs, and move them to separate frames.

To detach an editor tab, do one of the following

- Drag this tab outside of the main window. A preview thumbnail appears:



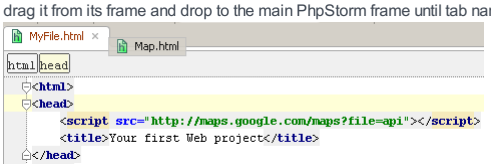
- Press `Shift+F4`.



The content of the editor tab opens in a separate frame.

To attach an editor tab

- drag it from its frame and drop to the main PhpStorm frame until tab name appears:



Tip The detached editor tabs can be split or unsplit.

- You can move editor tabs between split panes.

Editing Multiple Files Using Groups of Tabs

You can arrange tabs into groups to facilitate working with multiple files at a time. PhpStorm allows you to have an unlimited number of tab groups, thus enabling you to view several files or several places within the same file.

To create a new group of tabs

- Just split the desired editor tab.

To move a tab from one group to another

1. Switch to the desired tab.
2. Right-click the desired editor tab and choose Move Tab to Opposite Tab Group on the context menu.

Changing Placement of the Editor Tab Headers

By default, the tab headers appear on top of the editor, but you can change their location as required, and have the headers at the bottom, left, or right sides of the editor.

Note that the tab header placement is a global setting that applies to all projects.

To change location of the editor tab headers, do one of the following

- In the Editor Tabs settings page, select editor tab headers position from the Placement drop-down list.
- Right-click an editor tab, point to the menu item Tabs Placement, and then select the desired placement in the sub-menu.

Sorting Editor Tabs

On this page:

- Overview
- Enabling alphabetical sorting
- Sorting editor tabs alphabetically

Overview

The alphabetical sorting of the editor tabs is available regardless of the tabs position. However, alphabetical order for top and bottom placement becomes available, when the Show tabs in single row check command is selected on the Window | Editor Tabs | Tabs Placement menu.

Enabling alphabetical sorting

To enable alphabetical sorting:

1. On the main menu, choose Window menu.
2. Point to Editor Tabs | Tabs Placement.
3. Select the check command Show Tabs in Single Row.

Sorting editor tabs alphabetically

To sort editor tabs alphabetically:

1. Right-click an editor tab.
2. Select the check command Sort Tabs by Filename.
   If this check command is selected, the tabs headers are presented in alphabetical order. Otherwise, the editor tab headers appear in the opening order of the corresponding files.

Using TODO

In this section:

- Using TODO
  - Introduction
- Defining TODO Patterns and Filters
- Creating TODO Items
- Viewing TODO Items

Introduction

Working on a large project, you often need to create the lists of tasks, and keep your team mates informed about the issues that require their attention. Such issues can include the questions that should be answered, certain changes that should be done later, areas of optimization and improvement etc.

PhpStorm suggests to use special TODO comments in the source code. Such comments can be used in all supported file types, and should match a certain TODO pattern. PhpStorm comes with one pre-defined pattern, but you can define as many TODO patterns as required. When a matching occurrence is encountered, it is interpreted as a TODO item. PhpStorm highlights such comments in accordance with the Colors and Fonts settings.

Defining TODO Patterns and Filters

On this page:

- Basics
- Defining TODO patterns
- Defining filters

Basics

TODO items in the source code are defined by a certain pattern.

Whenever a pattern is changed, or a new pattern is added, PhpStorm scans the whole project and rebuilds the index of TODO items. Results display in the TODO tool window , as described in the section Viewing TODO Items .

By default, PhpStorm provides two patterns:

- `\btodo\b.*`
- `\bfixme\b.*`

A generic pattern looks like `todo.*`

You might want to view the TODO comments of certain a type, and hide the others. For this purpose, PhpStorm suggests to use filters. This way you can show those items that match certain patterns only.

Defining TODO patterns

To define a TODO pattern, follow these general steps

1. Open the TODO page of the Settings dialog.
2. In the Patterns section, click the Add button + to create a new pattern, or the Edit button to update an existing one. The Add/Edit Pattern dialog opens.
3. In the Pattern field, enter the regular expression that describes the desired pattern.
4. In the Icon list, select the desired icon that will mark the matching TODO items in the TODO tool window .
5. Specify the color and font properties, which PhpStorm will use to highlight the matching comments in the source code.
6. Select the Case sensitive check box, if you want the pattern to be case-sensitive.

Defining filters

To define a filter that will be used to show specific types of TODO items, follow these general steps

1. Open the TODO page of the Settings dialog.
2. In the Filters section, click click the Add button + to create a new filter, or the Edit button to update an existing one.
3. In the Add/Edit Filter dialog, specify the filter name, and select the patterns to be included in the filter.

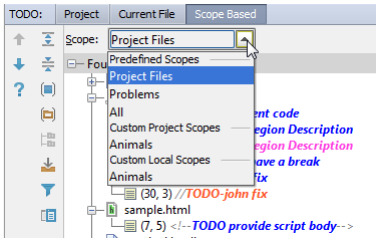
To create TODO items

1. Open the desired file in the editor and position the caret at the place where a TODO item should be created.
2. Create a comment . For example, you can use the `Ctrl+Slash` keyboard shortcut.
3. In the comment, type the string that matches one of your TODO patterns. By default, any string that starts with `TODO` (regardless of the case) is interpreted as a TODO item and is highlighted accordingly.
4. View the list of TODO items in the TODO tool window.

To view TODO items in project, follow these general steps

1. Open TODO tool window, as described in the procedure Showing a tool window .
   The tool window displays the encountered TODO items in several tabs:
   - All over the project (Project tab)
   - In the file currently active in the editor(Current File tab)
   - In one of the already defined scopes (Scope Based tab), which is quite useful for large projects.
   - In the current changelist, if version control support is enabled.

2. Click the desired tab (view), and explore the list of encountered TODO items. For example, with the Scope Based view selected, one has to choose scope from the drop-down list.



3. Narrow down the list of search results by choosing scope, and applying filters.

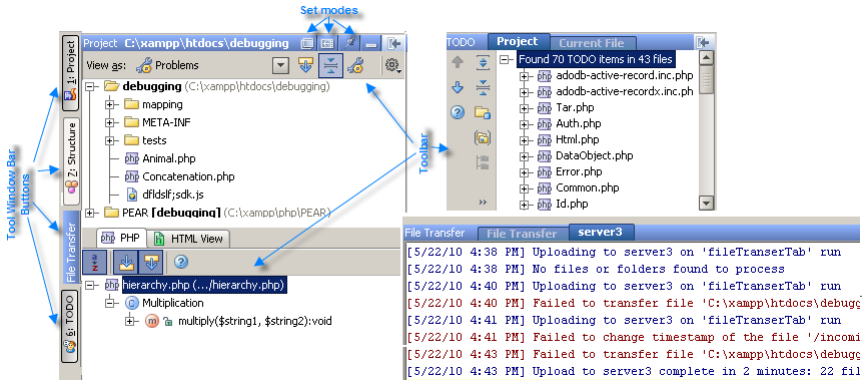
PhpStorm Tool Windows

In this section:

- PhpStorm Tool Windows
  - Overview
  - Tool window quick access
  - Tool window bars and buttons
  - General tool window layout
  - Accessing tool window menus
- Manipulating the Tool Windows
- Specifying the Appearance Settings for Tool Windows
- Speed Search in the Tool Windows
- Viewing Modes
- Managing Your Project Favorites

Overview

Attached to the bottom and sides of the workspace are PhpStorm tool windows. These secondary windows let you look at your project from different perspectives and provide access to typical development tasks. These include project management, source code search and navigation, running and debugging, integration with version control systems, and many other specific tasks.

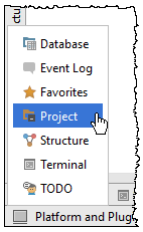


Certain tool windows are available always, that is, in any project irrespective of the project nature, contents, and configuration. Other tool windows are available only if the corresponding plugins are enabled. There are also tool windows that require certain specific actions to be made to become available. (For example, to make the Find tool window available you have to search for something first.)

Tool window quick access

In the lower left corner of the workspace, there is a button which initially looks like this .

If you point to this button, a menu opens that provides quick access to tool windows.

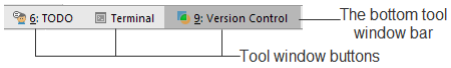


Option names in this menu correspond to the names of the tool windows. When you select an option, the corresponding tool window opens and becomes active.

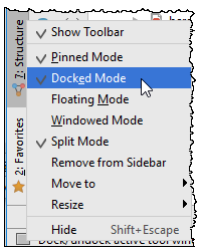
If you click the button, tool window bars and buttons are shown. At the same time the button appearance changes to . If you click the button again, the tool window bars and buttons are again hidden.

Tool window bars and buttons

When visible, the tool window button bars (or just tool window bars) are around the tool windows (or the editor area if the tool windows are hidden). These bars contain the buttons for showing or hiding the tool windows (tool window buttons).



The tool window buttons also provide access to tool window context menus which are shown when the buttons are right-clicked.



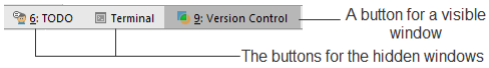
The context menus let you control the tool window viewing modes and other aspects of the tool window appearance.

Initially, there are three button bars, two at the sides of the main window and one at the bottom. You can show or hide all the button bars at once by clicking in the bottom-left corner of the workspace.

Each tool window button has the name of the corresponding tool window on it. On certain buttons, the window name may be preceded by a number, for example, 1: Project. This means that the keyboard shortcut `Alt+<number>` is available for showing or hiding the window. You can, for example, show or hide the Project tool window by pressing `Alt+1` .

You can turn showing the window access numbers on the buttons on and off in the Appearance settings .

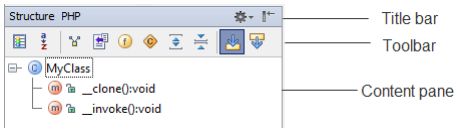
The buttons for visible tool windows and for hidden ones are shown differently.



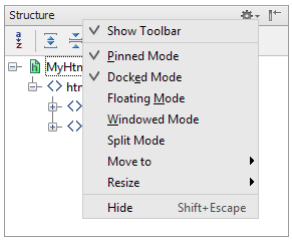
You can rearrange the tool windows by dragging-and-dropping the tool window buttons onto a different tool window bar (or to a different corner of the same bar). As a result, the tool window becomes attached to the bar you've moved the window button to.

General tool window layout

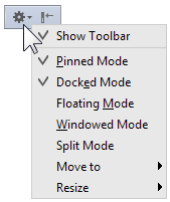
Generally, all the tool windows are organized in a similar way.



At the top of the window is a title bar. When you right-click the title bar, a menu for managing the window appearance and contents is shown.



The title bar contains two buttons in its right-hand part. The first of these buttons ( ) opens a menu for managing the tool window viewing modes . Note that the menu options are a subset of the title bar context menu.



The second of the buttons ( ) is for hiding the tool window. When used in combination with the `Alt` key, clicking this button hides all the windows attached to the same tool window bar.

Underneath the title bar are the toolbar and content pane. Depending on the window, the toolbar may be above or to the left of the content pane.

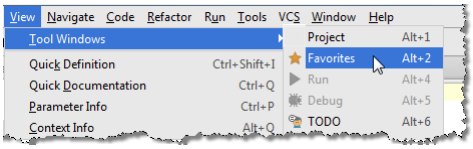
The toolbar buttons, generally, are window-specific. However, the windows with similar purposes may contain similar controls on their toolbars.

In most cases, a function associated with a toolbar button may also be accessed from the main or context menu, or may have a keyboard equivalent.

The content panes may be plain or contain two or more "layers" (views) represented, for example, by tabs. There are also tool windows with the content pane part shown on a separate tab in the editor area.

Accessing tool window menus

– Use the View | Tool Windows menu to show or hide the tool windows.



Warning! Note that some of the tool windows (for example, Find tool window) appear in the View menu ONLY when an action has been performed for the first time. So, unless you do something, say, find these tool windows are not visible in the View menu.

Manipulating the Tool Windows

You can manipulate the tool windows in various ways to adjust PhpStorm workspace to your needs.

- Showing a tool window
- Hiding an individual tool window
- Hiding all tool windows attached to the same tool window bar
- Hiding all tool windows
- Switching to the last active tool window
- Hiding or showing the tool window bars
- Hiding tool window buttons
- Attaching a tool window to a different tool window bar
- Resizing a tool window
- Increasing the number of tool windows shown at a time
- Saving and restoring the arrangement of the tool windows

Showing a tool window

Do one of the following:

- In the lower left corner of the workspace, point to and select the tool window in the menu that is shown.
- Click the corresponding tool window button on the tool window bar .
- Choose View | Tool Windows | <tool window> in the main menu.
- If the tool window has an associated quick access number, press `Alt+<number>` (for example, `Alt+1` for the Project tool window).

Hiding an individual tool window

Do one of the following:

- Click the corresponding tool window button on the tool window bar .
- Click on the tool window title bar.
- Right-click the corresponding tool window button and select Hide from the context menu.
- Right-click the tool window title bar and select Hide from the context menu.
- Choose View | Tool Windows | <tool window> in the main menu.
- If the tool window has an associated quick access number, press `Alt+<number>` (for example, `Alt+1` for the Project tool window).
- If the tool window you are going to hide is currently active, press `Shift+Escape`.

Hiding all tool windows attached to the same tool window bar

Do one of the following:

- Press and hold the `Alt` key, and click on the title bar of any of the tool windows attached to the corresponding tool window bar.
- Choose Window | Active Tool Window | Hide Side Tool Windows in the main menu. This command hides all the tool windows attached to same tool window bar as the active tool window or the last of the active tool windows.

Hiding all tool windows

Do one of the following:

- Choose Window | Active Tool Window | Hide All Windows in the main menu.
- Press `Ctrl+Shift+F12`.

Switching to the last active tool window

Do one of the following:

- Choose Window | Active Tool Window | Jump to Last Tool Window in the main menu.
- Press `F12`.

If all the tool windows are currently hidden, the last active tool window will be shown and made active.

Hiding or showing the tool window bars

You can hide all the tool window bars if you need more space in the PhpStorm window:

- In the lower left corner of the workspace, click .

If the tool window bars are hidden, you can bring them back onto the screen either permanently or for a short period of time:

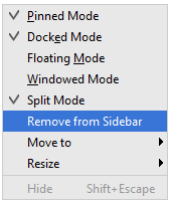
- To restore the tool window bars, click in the lower left corner of the workspace.
- To see the tool window bars for a short period of time, double-press and hold the `Alt` key. The tool window bars appear on the screen making the tool window buttons accessible. The tool window bars are hidden again when you release the key.

Hiding tool window buttons

PhpStorm makes it possible to hide individual tool window buttons, without actually uninstalling the corresponding plugins.

To remove a tool window button from view:

1. Make sure that the tool window bars are visible .
2. Right-click the tool window button you want to hide.
3. Choose Remove from Sidebar



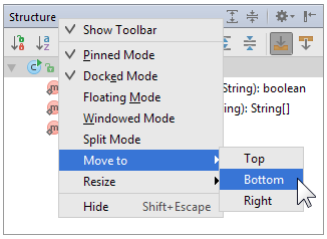
To restore the hidden tool window button, choose View | Tool Windows on the main menu, and then click the window with the hidden toolbar button.

Attaching a tool window to a different tool window bar

Do one of the following:

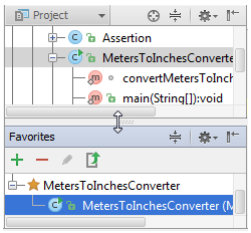
- Drag the corresponding tool window button onto the desired tool window bar (top, left, bottom or right).

- Right-click the corresponding tool window button or the tool window title bar to open the context menu. Choose Move to and then select the destination tool window bar (Top, Left, Bottom or Right).



Resizing a tool window

- Hover the mouse pointer over the tool window border. When the pointer becomes a double-headed arrow, drag the border in the required direction.



- You can also resize a tool window by moving its border to left or right, or up or down in steps. The following alternatives are available for doing that:
  - Right-click the corresponding tool window button or title bar and select Resize. Then select one of the available Stretch to options.
  - Make the tool window of interest active and do one of the following:
    - Choose Window | Active Tool Window | Resize, and then select the necessary Stretch to option.
    - Press `Ctrl+Shift` in combination with the corresponding arrow key.

Increasing the number of tool windows shown at a time

To increase the number of tool windows to be shown at a time, you should appropriately set the viewing modes for different tool windows. Consider the following:

- Generally, for a tool window to be visible always (i.e. even when inactive), the tool window should be pinned .
- There are no limiting factors for the number of visible floating windows and ones in the windowed mode. Note that the windowed mode is not available if you are using OS X.
- To be able to see two windows docked to the same tool window bar at a time (rather than one), one of the windows should have the split mode off and the other one on.
- Initially, three (out of four) tool window bars are used. You can "activate" the forth tool window bar (the top one) by moving certain tool windows to it.

Saving and restoring the arrangement of the tool windows

You can save the way the tool window are currently arranged by choosing Window | Store Current Layout as Default in the main menu.

At a later time, you can return to the saved workspace layout by choosing Window | Restore Default Layout (`Shift+F12`).

Specifying the Appearance Settings for Tool Windows

You can change certain tool window appearance properties by specifying the corresponding Appearance settings.

To change the appearance properties for tool windows

1. In the Settings dialog, expand the Appearance&Behaviour node, and click Appearance .
2. If necessary, change the settings related to tool window appearance. These are mainly in the Transparency and the Window Options sections. For more information, see descriptions of the pages under Appearance and Behavior .

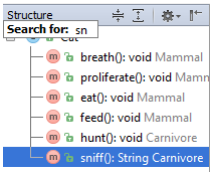
Speed Search in the Tool Windows

Speed search in the tool windows helps you find and navigate to a file or folder in the Project tool window, a member in the Structure tool window, a changelist in the Version Control tool window, an item in the TODO list, and more.

Note that speed search is performed only on epanded nodes, if a node is folded the matching items under it are not detected.

To search through a tool window

1. Select the desired tool window.
2. Start typing the item name (for instance, file, class, field, etc.). As you type, the Search for field appears over the tool window toolbar showing the entered characters, and the element selection moves to the first item that matches the specified string. The matching part of the string is highlighted.



3. If several neighboring items match the pattern, use the Up and Down keys on the keyboard to navigate among them.
4. Press `Enter` when ready. As a result, the matching item is selected in the tool window. Pressing `Escape` hides the Search for field.

Viewing Modes

PhpStorm provides various viewing modes that let you control the way the tool windows are shown and behave. These modes help you keep a proper balance between quick, easy access to tool windows and maximum screen space for editing your code.

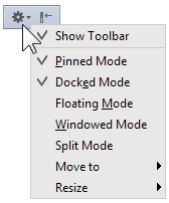
The viewing modes are set separately for each of the tool windows.

- Ways to control the viewing modes
- Fixed / floating / windowed mode
- Docked / undocked mode
- Pinned / unpinned mode
- Split mode
- Group Tabs option
- Wide screen support

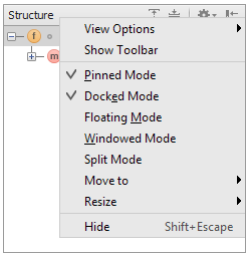
Ways to control the viewing modes

The viewing modes are set by turning the corresponding viewing options on or off. To access these options, you can use:

- on the title bar of a tool window.



- Context menus. The context menus are accessed by right-clicking the tool window buttons or the tool window title bars.

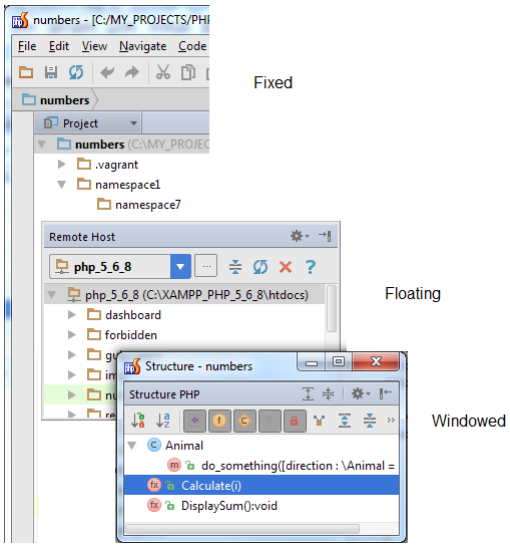


- For an active tool window: the Window | Active Tool Window menu.

Fixed / floating / windowed mode

A tool window may be fixed. In that case, it stays within the main window.

Alternatively, a tool window may be in the floating or in the windowed mode. Note that the windowed mode is not available if you are using OS X.



When in the fixed mode, one side of the tool window is attached to one of the tool window bars. The behavior of the opposite side depends on whether the window is docked or undocked .

Initially, all the tool windows are in the fixed mode (i.e. the floating and windowed modes are off).

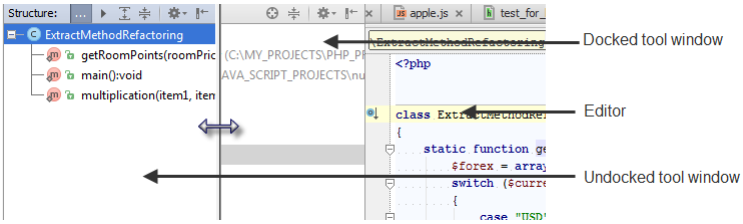
In the floating and in the windowed modes, a tool window may be moved around the screen to any place.

To switch to the floating or the windowed mode, turn on the Floating Mode or the Windowed Mode option. To bring a tool window back to the fixed mode, turn the Floating Mode and the Windowed Mode options off. See Ways to control the viewing modes.

Note that for the tool windows in the windowed mode, the Window menu command Hide Active Tool Window is disabled.

Docked / undocked mode

A tool window in the fixed mode may be docked or undocked.



In the docked mode, all the sides of a tool window are attached to surrounding elements (the editor, other tool windows, etc.) Thus, the tool window and the adjacent elements share the space available in the main window.

When a docked tool window becomes inactive, it stays visible or is hidden depending on whether the window is pinned or unpinned .

Initially, all the tool windows are in the docked mode (i.e. the docked mode is on).

When undocked, all the sides of a tool window (except the one at the tool window bar) are detached from surrounding elements. The window moves to the "upper layer" covering the elements it used to share the space with. In one of the directions (along the tool window bar), the window stretches and takes all the available space. In the other direction, one of the window borders becomes loose and can be moved without affecting the sizes of other, underlying elements.

When an undocked tool window becomes inactive, it is automatically hidden.

To switch between the docked and undocked mode, turn the Docked Mode option on or off. See Ways to control the viewing modes.

Pinned / unpinned mode

Pinned tool windows, generally, stay visible when becoming inactive. Unpinned tool windows in such cases are automatically hidden.

Initially, all the tool windows are pinned (i.e. the pinned mode is on).

There may be slight differences in behavior depending on the other viewing modes:

- Undocked tool windows are always hidden when inactive. (In the undocked mode, the tool windows are effectively unpinned.)
- Floating pinned tool windows, when inactive, may become semi-transparent.

To switch between the pinned and unpinned mode, turn the Pinned Mode option on or off. See Ways to control the viewing modes.

Split mode

This mode has to do with how many windows docked to the same tool window bar may be shown at a time (one or two).

Generally, the space along a tool window bar is shared between two groups of docked tool windows.

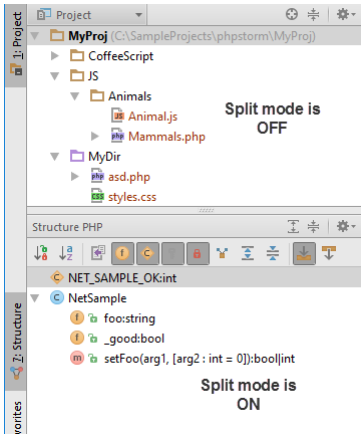
In one of the groups are the tool windows for which the split mode is off; in the other group are the ones with this mode on.

At each moment of time, only one window from each of the groups may be visible.

Thus, if all the tool windows docked to a tool window bar have the split mode off, only one tool window may be shown at a time. In this case, the tool window which is visible takes all the space available along the tool window bar. So when you make a certain window visible, the previous visible window is automatically hidden.

The same behavior is observed if the split mode is on for all the windows docked to the same tool window bar.

To be able to see two windows simultaneously, the corresponding windows should belong to different groups, that is, one of the windows should have the split mode off and the other one on.



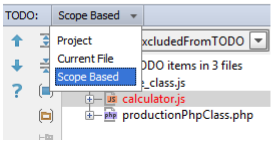
The tool window buttons for the tool windows with different settings of the split mode are grouped and shown at different corners of the corresponding tool window bar. For vertical window bars, the windows with the split mode off have the buttons at the top corner; for the horizontal bars, the buttons for such windows are at the left corner.

To turn the split mode on or off see Ways to control the viewing modes.

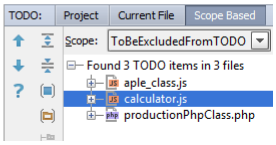
Group Tabs option

If more than one view is available in a tool window, the corresponding views may be shown on separate tabs if the Group Tabs option is off. If this option is on, the views are selected from a list.

Group Tabs: ON



Group Tabs: OFF



Wide screen support

PhpStorm makes it possible for the tool windows to use the full width and height of the screen. In the Settings dialog, expand the node Appearance and Behaviour, and in the Appearance page, use the check boxes Wide screen tool window layout and Side by side layout on the left/right to optimize placement of the tool windows.

Note also that you can turn side-by-side layout on or off by `Ctrl+MouseClick` on splitter between the tool windows.

Refer to Appearance for details.

Managing Your Project Favorites

On this page:

- Basics
- Using the Project tool window to add items to favorites
- Using the editor to add items to favorites
- Creating a new favorites list
- Renaming a favorites list
- Moving an item to a different list
- Removing items from favorites

Basics

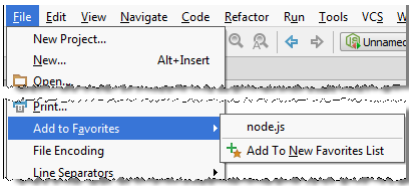
You can arrange the most frequently used files, folders, bookmarks , and breakpoints in the lists of favorite items (favorites). In PhpStorm, there is a dedicated tool window for managing your favorites (the Favorites tool window).

Initially, there is one (empty) favorites list which has the same name as the project.

You can create more favorites lists and manage their contents as necessary.

Using the Project tool window to add items to favorites

1. In the Project , select the item or items you want to add to favorites.
2. Do one of the following:
   - On the main menu, point to File | Add To Favorites.
   - On the context menu of the selection, point to Add To Favorites
3. To add the selected item or items to an existing favorites list, select the name of the list.
   To create a new favorites list and add the selected item or items to it, select Add To New Favorites List. In the Add New Favorites List dialog, specify the name of the new list and click OK.



Tip Add items to favorites using drag-and-drop: drag the item of interest from the Project tool window, or an external file from Explorer or Finder, and drop it onto the desired favorites list in the Favorites tool window.

Using the editor to add items to favorites

In the editor, you can add to favorites one file, or all the currently opened files.

1. Right-click the editor tab of interest and select one of the following options:
   - If you want to add the current file to favorites, select Add To Favorites. (For the current file, File | Add To Favorites is also available as an alternative.)
   - If you want to add all the files open in the editor to favorites, select Add All To Favorites.
2. To add the item or items to an existing favorites list, select the name of the list.
   To create a new favorites list and add the item or items to it, select Add To New Favorites List. In the Add New Favorites List dialog, specify the name of the new list and click OK.

Creating a new favorites list

You have an option of creating a new favorites list when adding items to favorites or when moving a favorites list item to a different list.

You can also create a new (empty) favorites list just on its own, as a separate task:

1. If the Favorites tool window is not currently shown, open it .
2. Do one of the following:
   - Click + on the toolbar.
   - Press `Alt+Insert`.
3. In the Add New Favorites List dialog, specify the name of the new list and click OK.

Renaming a favorites list

1. If the Favorites tool window is not currently shown, open it .
2. Do one of the following:
   - Right-click the list whose name you want to change and select Rename Favorites List.
   - In the toolbar of the Favorites tool window, click .
3. In the New Name for Favorites List dialog box, change the name of the list as required, and click OK.

Moving an item to a different list

1. If the Favorites tool window is not currently shown, open it .
2. Right-click the list item that you are going to move and select Send To Favorites.
3. To move the selected item to an existing favorites list, select the name of the destination list.
   To create a new favorites list and move the item there, select Send To New Favorites List. In the Add New Favorites List dialog, specify the name of the new list and click OK.

Tip To move an item from one favorites list to another, use drag-and-drop.

Removing items from favorites

To remove items from favorites, you can delete the corresponding favorites list items and/or the whole favorites lists.

1. If the Favorites tool window is not currently shown, open it .
2. Select the item or items that you want to remove from favorites. Note that you can select separate list items and the whole lists at the same time.
3. Do one of the following:
   - Click on the toolbar.
   - Select Remove From Favorites in the context menu. (If a single favorites list is currently selected, note that there are also the following options: Delete Favorites List <list_name> and Delete All Favorites Lists Except <list_name>.)
   - Press `Delete`.

Configuring Project and IDE Settings

In this section:

- Configuring Project and IDE Settings
  - Introduction
- Accessing Default Settings
- Accessing Settings
- Configuring Code Style
- Configuring Colors and Fonts
- Configuring Color Scheme for Consoles
- Configuring Encoding for Properties Files
- Configuring Individual File Encoding
- Configuring Keyboard Shortcuts
- Configuring Line Separators
- Configuring Menus and Toolbars
- Configuring Quick Lists
- Configuring Scopes and File Colors
- Configuring Third-Party Tools
- Copying Code Style Settings
- Creating and Registering File Types
- Exporting and Importing Settings
- Sharing Your IDE Settings
- Switching Between Schemes
- Switching Boot JDK

Note Find descriptions of the other specific configuration procedures in the relevant procedural parts.

Introduction

Project settings refer to a set of preferences related to resources, file colors, version control options, code styles, etc. Project settings are stored with each specific project as a set of `xml` files under the `.idea` folder.

You can configure project settings on the two possible levels:

- The level of a template project. The settings defined for a template project, apply to any project you create.
- The project level. The settings defined on this level apply to the current project only.

Accessing Default Settings

PhpStorm helps define settings of a default project. These settings are used as defaults every time you create a new project.

To access default project settings

1. On the main menu, choose File | Default Settings.
2. Define the desired settings in the Settings dialog box that opens.

Accessing Settings

On this page:

- Introduction
- Opening the Settings / Preferences dialog
- Finding an option or setting
- Finding an option or setting using Search Everywhere or Find Action

Introduction

This section describes simple steps required to access the Settings/Preferences dialog. Note that the settings that pertain to the current project, are marked with icon.

Opening the Settings / Preferences dialog

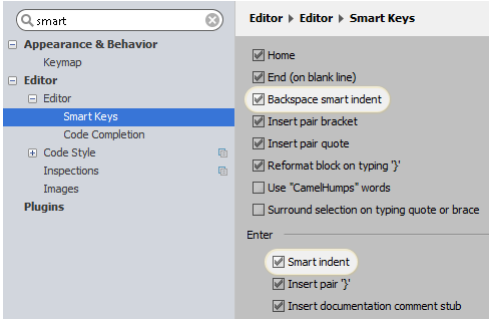
Do one of the following:

- Press `Ctrl+Alt+S`.
- On the main toolbar, click .
- On the main menu, choose File | Settings for Windows and Linux or PhpStorm | Preferences for OS X
- Press `Ctrl+Shift+A`, type `settings` and press `Enter`. See Navigating to Action .
- Click in the upper-right corner of the PhpStorm window, and type `#`.

Refer to Finding an option or setting using Search Everywhere or Find Action below.

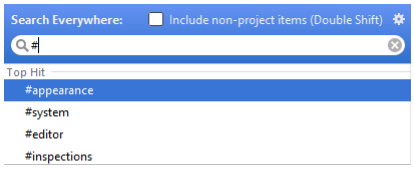
Finding an option or setting

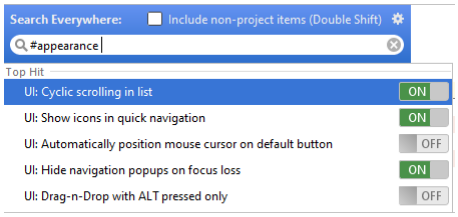
1. Open the Settings / Preferences dialog .
2. In the search field, start typing the text that you expect to find in the name of the setting. As soon as the specified text is found, the matching element is highlighted and the corresponding page is displayed.

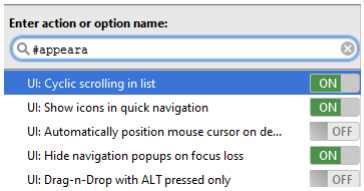


Finding an option or setting using Search Everywhere or Find Action

You can also use Searching Everywhere or Find Action . To find an option or setting, first type `#` character, and then choose one of the suggested categories:







Configuring Code Style

On this page:

- Basics and definitions
- Configuring code style for a language
- Copying code style settings from other languages
- Applying framework-specific pre-configured coding standards
- Configuring the code style for a project using EditorConfig

Basics and definitions

If certain coding guidelines exist in a company, one has to follow these guidelines when creating source code. PhpStorm helps maintain the required code style.

The code styles are defined at the project level and at the global level.

- At the Project level, settings are grouped under the Project scheme, which is predefined and is marked in bold font. The Project style scheme applies to the current project only.
  The default scheme cannot be changed! When any modifications are made, PhpStorm automatically creates a copy of the default scheme. If you want to use a different project level scheme, choose the desired scheme and copy its settings to project.
- At the Global level, settings are grouped under the predefined Default scheme, and any other scheme created by the user. The global settings are used when the user doesn't want to keep code style settings with the project and share them.

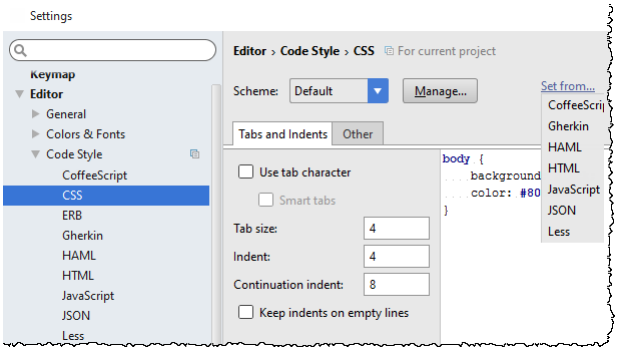
Configuring code style for a language

1. In Settings, click Code Style , and then click the language in question.
2. Choose the code style scheme to be used as the base for your custom coding style for the selected language.
3. Browse through the tabs of the selected language page, and configure code style preferences for it.

Copying code style settings from other languages

For most of the supported languages, you can copy code style settings from other languages or frameworks.

1. In Settings, click Code Style , and then click the language in question.
2. Click the link Set From in the upper-right corner. This link appears for those languages only, where defining settings on the base of the other languages is applicable.
3. In the drop-down list that appears, click the language to copy the code style from:



Applying framework-specific pre-configured coding standards

For PHP files, you can have framework-specific pre-configured coding standards applied.

1. In Settings, click Code Style , and then click the language in question.
2. Click the Set from link, choose Predefined, then choose the relevant pre-configured standard.

Configuring the code style for a project using EditorConfig

Before you start working with EditorConfig, make sure that the EditorConfig plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Make sure that the check box Enable EditorConfig Support is selected in Editor | Code Style.

For more information, see EditorConfig Website.

To configure the code style for a project using EditorConfig:

1. In the project tree, right-click a directory where you want to create the EditorConfig settings file and select New | File.
2. In the dialog that opens, enter `.editorconfig` and click OK.
   PhpStorm creates an EditorConfig settings file and displays a notification in the pop-up window.

   Every time you open a file, the EditorConfig plugin looks for a file named `.editorconfig` in the directory of the opened file and in every parent directory. A search for `.editorconfig` files will stop if the root file path is reached or an EditorConfig file with `root = true` is found. Therefore, if you want to use the IDE settings instead of the EditorConfig settings, clear the Enable EditConfig Support check box in Editor | Code Style that is selected by default.
3. Start defining your code style settings. Save(Ctrl+S) your file. Every time you modify the `.editorconfig` file, save the file to apply changes to your project.

   The EditorConfig code style configuration overrides the code style configuration in the IDE settings.

Configuring Colors and Fonts

On this page:

- Basics
- Configuring colors and fonts
- Example

Basics

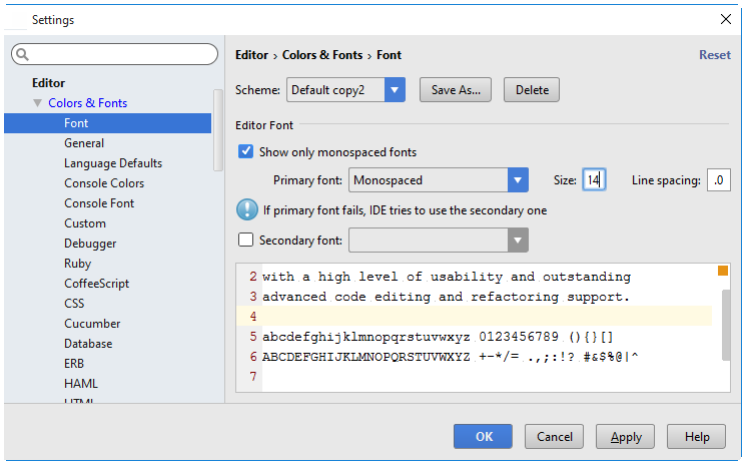
With PhpStorm, you can maintain your preferable colors and fonts layout for syntax and error highlighting in the editor, search results, Debugger and consoles via font and color schemes.

PhpStorm comes with a number of pre-defined color schemes. You can select one of them, or create your own one, and configure its settings to your taste. Note that pre-defined schemes are not editable. You have to create a copy of a scheme, and then change it as required.

Configuring colors and fonts

To configure color and font scheme

1. Open Settings, and under the Editor node, click Colors & Fonts .
2. Select the desired scheme from the Scheme name drop-down list.
3. If you need to change certain settings of the selected scheme, create its copy. To do that, click the Save as button, and type the new scheme name.
4. Under the Colors and Fonts node, change the font families used in the editor and in the console:
   - Define font family for the editor and console. When you just open the Font or Console Fonts pages under the Colors and Fonts node, PhpStorm displays Editor Font area where you can configure primary and secondary fonts, their size and line spacing.



5. Under the Colors and Fonts node, open pages to configure specific color preferences and font types for the different supported languages and PhpStorm components.

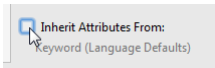
> **Tip** You can view how the new scheme looks in the editor. To do that, just click Apply without closing the Settings dialog box.

Example

Changing font for JavaScript

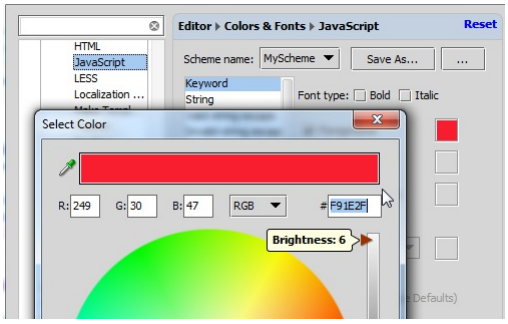
Click JavaScript node.

Important! Make sure the Inherit attributes from <component> check box is cleared:



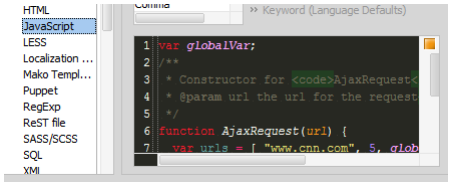
In the list of language components select Keyword.

Define the desired font type using the Bold and Italic check boxes. Next, select the Foreground check box, and right click the color preview to customize:



Refer to the description of the Color Picker dialog box for details.

Observe results in the preview pane:



Configuring Color Scheme for Consoles

PhpStorm enables you to define your habitual color scheme for the various types of consoles. So doing, you can individually configure all sorts of console output and user input.

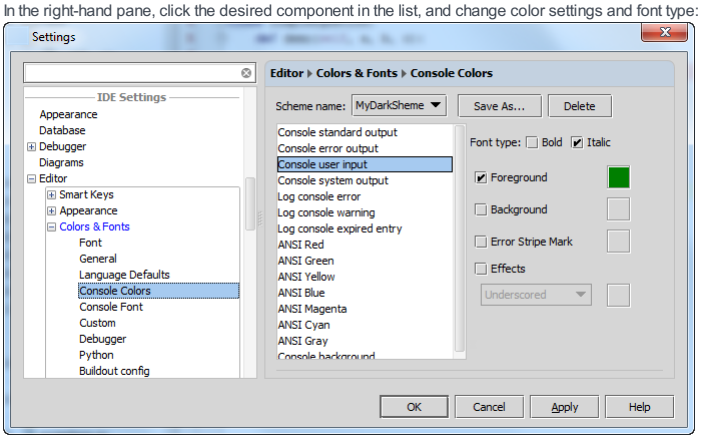
Color scheme includes numerous colors for background, user input, system output, and error output.

To configure color and font scheme for consoles

1. Make sure you are working with an editable scheme.

2. Open the IDE Settings , and under Colors&Fonts , scroll through the list of components, and select the ones related to consoles:
   - Console Colors
   - Console Fonts

3. In the right-hand pane, click the desired component in the list, and change color settings and font type:



Configuring Encoding for properties Files

PhpStorm applies system default encoding to the properties file.

To use other encodings, feed them as escape sequences and Unicode. The other option is to define the default encoding for `.properties` files on the project level and use different API that can read properties files in the encoding you have defined.

To configure default encoding for properties files

1. Open the Settings dialog box and click File Encodings .

2. Do one of the following:
   - To have the special mode turned on when symbols are stored in a properties file as escape sequences but displayed as normal letters, check the option Transparent native-to-ascii conversion. This option is helpful when the properties files are encoded in ISO 8859-1. It is recommended to use this approach, if you don't have any special reasons to change encoding.
   - In the field Default encoding for properties files, select the encoding that will be used for all properties files in project.

Configuring Individual File Encoding

On this page:

- Basics
- Configuring encoding for a directory or file without embedded encoding information
- Changing encoding of a file with explicit encoding
- Changing encoding of a file without explicit encoding

Basics

There are two modes of dealing with file encoding:

- Converting: the contents of the editor are stored in a different encoding. So doing, the contents of the underlying file change, but the contents of the editor stay unchanged.
- Reloading: the underlying file, opened in the editor, is shown in an encoding that differs from its original one. So doing, the contents of the editor can change, but the underlying files does not.

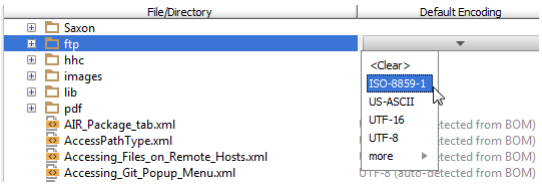
PhpStorm suggests the following major ways to change encoding:

- Using the File Encodings page of the Settings dialog, for directories and for the files that do not contain encoding information.
- Using the Status bar or menu command , for individual files that do not contain encoding information.
- Using the editor , for individual files that contain encoding information.

Configuring encoding for a directory or file without embedded encoding information

To configure encoding for a directory or file without embedded encoding information

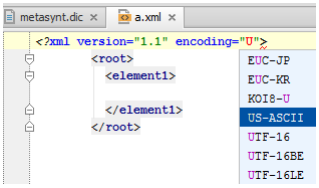
1. In Settings, expand the Editor node and select File Encodings .
2. The File/Directory column shows the tree view of your project. The Default Encoding column shows encoding for directories or files. Click the Default encoding column for a directory or file you want to define encoding for, and then choose the desired encoding from the drop-down list:



Changing encoding of a file with explicit encoding

To change encoding of a file that contains explicit encoding

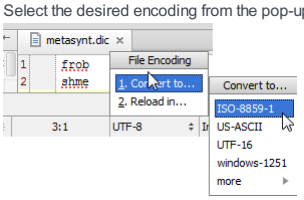
1. Open the desired file in the editor.
2. Change explicit encoding information. Use error highlighting to recognize wrong encoding and press `Ctrl+Space` to have a list of available encodings displayed:



Changing encoding of a file without explicit encoding

To change encoding of a single file that doesn't contain explicit encoding

1. Open the desired file for editing.
2. Do one of the following:
   - On the main menu, point to File | File encoding.
   - Click file encoding on the Status bar .
3. Select the desired encoding from the pop-up window.



4. If the selected encoding will change the file contents, PhpStorm shows a dialog box, where you can choose to Reload file from disk, or Convert it to a different encoding.

Configuring Keyboard Shortcuts

On this page:

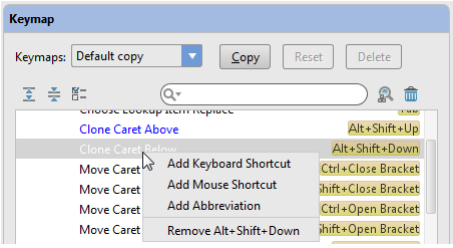
- Basics
- Where the keymaps are stored?
- Configuring keyboard shortcuts and mouse shortcuts
- Searching for actions

Basics

PhpStorm is a keyboard-centric IDE. Most of the actions (navigation, refactoring, debugging, etc.) can be carried out without using a mouse, which lets dramatically increase coding speed. If you had used another IDE for a while and have memorized your favorite keyboard shortcuts, you can use them all in PhpStorm.

PhpStorm completely suits your *shortcut habits* by supporting customizable keymaps. A keymap is a set of keyboard and mouse shortcuts that invoke different actions - menu commands, editor operations, etc. PhpStorm comes with a set of pre-configured keymaps.

Pre-configured keymaps are not editable. If you need to change some shortcuts, the copy of the current selected pre-defined keymap is created automatically:



Where the keymaps are stored?

All user-defined keymaps are stored in separate configuration files under the `config/keymaps` subdirectory in the PhpStorm profile directory:

- Windows and *NIX systems: `<User home>/.PhpStorm<xx>/config/keymaps`
- OS X: `~/Library/Preferences/PhpStorm<xx>/keymaps/`

Each keymap file contains only differences between the current and the parent keymaps.

Configuring keyboard shortcuts and mouse shortcuts

To configure keyboard shortcuts and mouse shortcuts

1. Open the Settings dialog box , and click Keymap .
2. Select one of the pre-configured Keymaps, which you want to use as the base for the new one, and click Copy. Accept the default name, or change it as required.
3. In the content pane of actions, select the desired action.
4. Configure keyboard shortcuts. To do that, follow these steps:
   1. Click on the toolbar, or right-click the selected action, and choose Add Keyboard Shortcut. Enter Keyboard Shortcut dialog box opens.
   2. Press the keys to be used as shortcuts. The keystrokes are immediately reflected in the First Stroke field. Optionally, select the check box next to Second Stroke and press keys to be used as alternative keyboard shortcuts.

      As you press the keys, the Preview field displays the suggested combination of keystrokes, and the Conflicts field displays warnings, if some of the keystrokes are already assigned to the other actions.
   3. Click OK with the mouse pointer to create a shortcut and bind it with an action.

   It is important to use the mouse pointer, because any keystroke is interpreted as a shortcut.
5. Configure mouse shortcuts. To do that, follow these steps:
   1. Click on the toolbar, or right-click the selected action, and choose Add Mouse Shortcut on the context menu, if you need to bind an action to a mouse click. Enter Mouse Shortcut dialog box opens.
   2. In the Click Count section, click a radio button to choose a Single Click or Double Click.
   3. Hover your mouse pointer over the section Click Pad and click the desired mouse button. Use `Alt` , `Ctrl` , and `Shift` modifiers for diversity. As you click, the Shortcut Preview field displays the suggested shortcut, and the Conflicts field displays warnings, if some of the shortcuts are already assigned to the other actions.
   4. Click OK or Press `Enter` to create a shortcut and bind it with an action.

If a conflict is reported, a warning message appears. You can choose one of the following options:

- Remove to remove all other bindings and preserve the new one.
- Leave to preserve all bindings including the new one.
- Cancel to return to the keymap definition.

Although you can ignore conflict and bind a shortcut with several actions, it is strictly recommended to avoid binding two actions with the same shortcut, because the order of performing such actions is not defined.

Searching for actions

Tip Use the Keymap page to search for certain actions by name, or by shortcut.

To find an action by name, type this name in the search field

. As you type, the content pane shows actions with the matching names.

To find an action by shortcut, click . In the Filter Settings dialog box, start pressing keys. The content pane shows only the actions with the matching shortcuts.

Click your mouse somewhere outside the Filter Settings dialog box to close it. Avoid using keys, since any keystroke is interpreted as a shortcut.

Configuring Line Separators

PhpStorm makes it possible to set up line separators (line endings) for the newly created files, and change line separator style for the existing files.

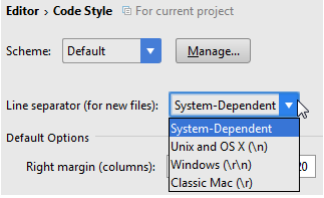
On this page:

- Setting up line separators for newly created files
- Viewing line ending style for the current file
- Changing line separator for a file, currently opened in the editor
- Changing line separator for a selection in the Project view
- Tips and tricks

To set up line separators for new files

1. In Settings, click Code Style .

2. From the Line separator (for new files) drop-down list, select the desired line separator style:



3. Apply changes and close the dialog.

To view line ending style for the current file

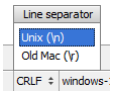
1. Open the desired file in the editor, as described in the section Opening and Reopening Files .

2. View the Status bar : the current line endings style is denoted by the dedicated icon with the specified style, for example, CRLF .

To change line separator for a file, currently opened in the editor

1. Open the desired file in the editor, as described in the section Opening and Reopening Files .

2. Do one of the following:
   - Click the line separator spin box in the Status bar , and choose the desired line ending style from the pop-up menu:

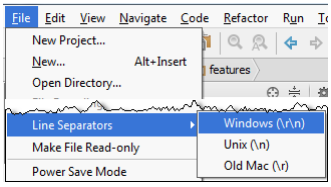


   - Choose File | Line Separators on the main menu, and then choose the desired line ending style from the sub-menu.

To change line separator for a file or directory, selected in the Project view

1. Select a file or directory in the Project tool window .
   Note that if a directory is selected, the line ending style applies to all nested files recursively.

2. Choose File | Line Separatorson the main menu, and then select the desired line ending style from the sub-menu.



Tips and tricks

- Use multiple selection in the Project view.
- Changing line separator is reflected in the Local history of a file.
- Run the inspection 'Inconsistent line separators' to find out, which files use line separator different from project's default.

Configuring Menus and Toolbars

You can customize menu and toolbar command lists to regroup features or make your favorites easier to access.

To customize menus and toolbars

1. Open Settings and click Menus and Toolbars . Alternatively, right-click the main toolbar, and choose Customize Menus and Toolbars on the context menu.

2. In the list of available menus and bars, expand the node you want to customize and select the desired item.

3. Customize the list of items in the selected menu or bar using the buttons on the right from the list:
   - To add a new command, select the desired location in the list and click the Add After button. In the Choose Action To Add dialog box that opens, select the desired action.
     Optionally associate the action with an icon using the Icon Path text box. In this text box, specify the location of the file with the icon you want to assign to the selected action. If necessary, use the Browse button ... to select the file in the corresponding dialog .

     Tip The image file should have .png extension.
     - The size of the toolbar icons should be 16x16.

   - To change the icon associated with a command, select the desired command in the list and click the Edit Action Icon button. In the Choose Actions Icon Path dialog box that opens, specify the location of the desired image. If necessary, use the Browse button ... to select the image in the Select Path Dialog .
   - To delete an item from the list, select it and click the Remove button.
   - To have logical groups of commands separated from each other by a separator, select the desired location in the list and click the Add Separator button.
   - To change the order in which commands appear in the selected menu or on the selected bar, use the Move Up and Move Down buttons.

4. To abandon the changes and return to the default settings, click the Restore Default button.

Configuring Quick Lists

On this page:

- Introduction
- Using a quick list
- Configuring a quick list

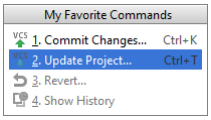
Introduction

A Quick List is a pop-up menu of PhpStorm commands, configured by the user and associated with a keyboard or mouse shortcut. You can create as many quick lists, as necessary. Each command, included in a quick list, is identified by a sequential number. Numbering starts from the numerals (0 to 9), and then proceeds with the letters in alphabetical order.

Using a quick list

To invoke a command from a quick list

1. Invoke quick list by its keyboard shortcut.
2. Select the desired command, using its number, the mouse cursor, or navigation keys and the `Enter` key:



Configuring a quick list

To configure a quick list

1. Open Settings/Preferences dialog, and click Quick Lists page.
2. Click + to create a new quick list.
3. In the Display name field, specify the name of the quick list. Optionally, provide the quick list description.
4. Configure the quick list. Use:
   - Add to add actions to the list. Select the actions in the Add Actions to Quick List dialog that opens.
   - Add Separator to add a separator at the end of the list.
   - Move Up and Move Down to move the selected item one line up or down in the list.
   - Remove to remove the selected item from the list.
5. Apply the changes.
6. Bind the new quick list with one or more shortcuts:
   - In the Keymaps page of the Settings/Preferences dialog, expand the Quick Lists node and select the new quick list.
   - Perform the key binding procedure . Note that you can only modify a custom keymap.
7. Apply the changes and close the dialog.

Configuring Scopes and File Colors

This section describes how to configure scopes and coloring of the files belonging to these scopes:

- Creating a new custom scope
- Configuring the list of items in a custom scope
- Associating file color with a scope
- Arranging the order of scopes

Project scopes are configured in the Scopes page of the Settings dialog box.

To create a new custom scope

1. In the Scopes settings page, click Add scope.
2. Select Local Changes or Shared from the drop-down list. Shared scopes are defined for the current project and accessible for the team members via VCS, while Local scopes are intended for personal use only and are stored in your workspace.

   Tip You can change the sharing state later using the Share scope check box in the bottom of the page.

3. Specify the name for the new scope.
4. Apply changes.

To configure the list of items in a custom scope

1. In the Scopes settings page, select the scope that you want to configure.
2. Do one of the following:
   - Choose files and folders to be included in the scope and use buttons on the right. Based on the inclusion/exclusion, PhpStorm creates an expression and displays it in the Pattern field.
   - Specify pattern in the Pattern field manually, using the scope language syntax .
3. Apply changes.

Files belonging to different scopes can be highlighted in different colors throughout the PhpStorm's user interface: in navigation lists , in the editor tabs, in the Project window . This allows much faster and easier navigation in large projects.

To associate file color with a scope

1. Open the File Colors settings page.
2. Decide whether you want the scope-color association to be only available to you or to be shared with the team. Depending on that, use either Local colors or Shared colors section of the page.

   Tip You can change the sharing state later by selecting a color in the list ans using the Share button in the Local colors section or the Unshare button in the Shared colors section of the page.

3. Click Add.
4. In the dialog box that opens, select a scope and pick a color for it.
5. If necessary, use the check boxes on top of the page to define where in the user interface files belonging to the scopes are highlighted.
6. Apply changes.

If some file is included into several scopes, the order of the scopes becomes important: PhpStorm uses the color of the uppermost scope (shown in the Scopes settings page) to highlight such file. Of course, you can change the order of the scopes, and thus the resulted highlighting.

To arrange the order of scopes

1. Open the Scopes settings page.
2. Select a scope whose position in the order you want to change.
3. Click Move Up / Move Down
4. Apply the changes.

Configuring Third-Party Tools

You can define third-party standalone applications (code generators and analyzers, pre- and post- processors, database utilities, etc.) as external tools and then run them from PhpStorm.

You can pass contextual information (like the currently selected file, or your project source path) to the external tools, view the tool output, and more.

The tools are defined on the External Tools page in the Settings dialog and appear as commands in the Tools menu and in various context menus. They can also be assigned keyboard shortcuts (see Configuring Keyboard Shortcuts).

Copying Code Style Settings

On this page:

- Introduction
- Creating a global copy of a code style scheme
- Creating a project copy of a code style scheme

Introduction

You can define the code styles that differ from the pre-defined ones. These code style schemes are stored in XML files, in the `config/codestyles` folder under the user home directory.

You can use the created copy for modifying code styles, and for export.

If you select a code style scheme other than Project, then this code style will be saved for a project. Thus you can assign a global code style for each project.

Creating a global copy of a code style scheme

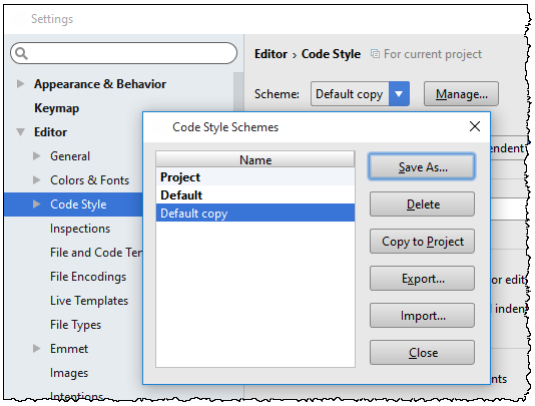
To create a copy of code style settings

1. In the Code Style page , select the desired scheme from the drop-down list, and click Manage.
2. In the Code Style Schemes dialog box that opens, click Save As....
3. In the Save Code Style Scheme As dialog box, type the name of the new scheme.
   PhpStorm saves the new code style with the specified name in the `config/codestyles/<code_style_name>.xml` file under the PhpStorm home directory.
4. Close the Code Style Schemes dialog box, and apply changes.

Creating a project copy of a code style scheme

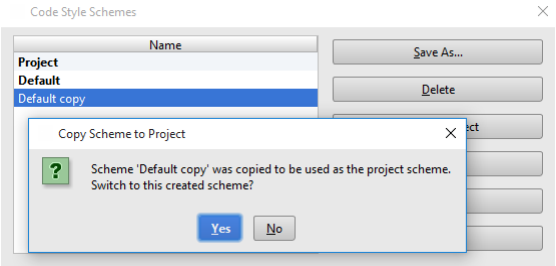
To copy code style settings to a current project

1. In the Code Style page , select the desired scheme from the drop-down list, and click Manage.
2. In the Code Style Schemes dialog box that opens, click Copy to Project:



   The selected code style is saved in the `.idea` directory in the file `codeStyleSettings.xml` .

3. In the Copy Scheme to Project dialog box, confirm switching to the new scheme, if necessary (Yes):



   Then close the Code Style Schemes dialog box, and apply changes.

Creating and Registering File Types

On this page:

- Introduction
- Creating a file type
- Registering a file type

Introduction

You can create custom file types to enable parsing these files in the editor by defining highlighting schemes for keywords, comments, numbers, etc. To enable PhpStorm to decide how to treat a file, you need to associate each file type with relevant extensions .

Creating a file type

To create a new file type

1. Open the Settings dialog by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Then select . Find more on page Accessing Settings .
2. On the File Types page that opens, click +.
3. In the New File Type dialog box that opens, specify the name of the new type and optionally provide a description.
4. In the Syntax Highlighting section, specify the characters for line and block comments, hex prefixes, and number postfixes.
5. In the Keywords section, specify sets of keywords using the tabs from 1 to 4. To do so, select the desired tab, click + (`Alt+Insert`), and enter the keyword name in the Add New Keyword dialog box that opens.

   Tip Each set of keywords has its own highlighting. You can change the highlighting color scheme for each set, on the Colors and Fonts page. Click the Custom tab and edit the Keyword1, Keyword2, Keyword3, and Keyword4 properties.

Registering a file type

To associate a file type with extensions

1. Open the File Types settings page.
2. From the Recognized File Types list, select the desired type.
3. In the Registered Patterns area, complete the list of patterns that define the file extensions to indicate that the corresponding files belong to the selected type. Do one of the following:
   - To register a new pattern, click + (`Alt+Insert`) and enter the desired extension pattern in the Add Wildcard dialog box that opens.
   - To update a pattern, select it in the list, click the Edit button and make the necessary changes in the Edit Wildcard dialog box that opens.
   - To remove a pattern from the list, select it and click – (`Alt+Delete`).

Exporting and Importing Settings

On this page:

- Introduction
- Exporting settings to a JAR archive
- Importing settings from a JAR archive

Introduction

PhpStorm enables you to preserve and share your working environment. You can archive and store your preferred IDE settings, put the settings file under version control and thus make it available to your colleagues. On the other hand, you can use the settings, defined by the other team members, or your own ones intended for a different usage.

Exporting settings to a JAR archive

To export IDE settings to a JAR archive

1. On the main menu, choose File | Export Settings.
2. In the Export Settings dialog box that opens specify the settings to export by selecting the check boxes next to them. By default, all settings are selected.
3. In the Export settings to text box, specify the fully qualified name of the target archive. Type the path manually or click the Browse button and specify the target file in the dialog that opens .

Importing settings from a JAR archive

To import settings from a JAR archive

1. On the main menu, choose File | Import Settings.
2. In the Import File Location dialog box that opens select the desired archive.
3. In the Select Components to Import dialog box that opens specify the settings to be imported, and click OK. By default, all settings are selected.

Sharing Your IDE Settings

On this page:

- Introduction
- Prerequisites
- Configuring settings repository
- Authentication
- Configuring a read-only source

Introduction

PhpStorm allows you to share your IDE settings between different instances of PhpStorm (or other IntelliJ platform-based) products installed on different computers.

This is useful if you are using several PhpStorm installations, or want to implement the same settings among your team members or company-wide.

Prerequisites

Before you start working with Settings Repository, make sure that the Settings Repository plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Configuring settings repository

If you want to share your IDE settings, perform the following steps:

1. Create a repository on GitHub .
2. On the computer where the PhpStorm instance whose settings you want to share is installed, navigate to File | Settings Repository. Specify the URL of the repository you've created and click Overwrite Remote.
3. On each computer where you want your settings to be applied, in the Settings/Preferences dialog , expand the Tools node and choose Settings Repository , specify the URL of the repository you've created, and click Overwrite Local.
   You can click Merge if you want the repository to keep a combination of the remote settings and your local settings. If any conflicts are detected, a dialog will be displayed where you can resolve these conflicts.

   If you want to overwrite the remote settings with your local settings, click Overwrite Remote.

Your local settings will be automatically synchronized with the settings stored in the repository each time you perform an Update Project or a Push operation, or when you close your project or exit PhpStorm.

If you want to disable automatic settings synchronization, navigate to File | Settings | Tools | Settings Repository and disable the Auto Sync option. You will be able to update your settings manually by choosing VCS | Sync Settings from the main menu.

Authentication

On the first sync, you will be prompted to specify a username and password.

It is recommended to use an access token for GitHub authentication. If, for some reason, you want to use a username and password instead of an access token, or your Git hosting provider doesn't support it, it is recommended to configure the Git credentials helper .

Note that the OS X Keychain is supported, which means you can share your credentials between all IntelliJ Platform-based products (you will be prompted to grant access if the original IDE is different from the requestor IDE).

Configuring a read-only source

Apart from the Settings Repository, you can configure any number of additional repositories containing any types of settings you want to share, including live templates, file templates, schemes, deployment options, etc.

These repositories are referred to as Read-only sources, as they cannot be overwritten or merged, just used as a source of settings as is.

To configure such repositories, do the following:

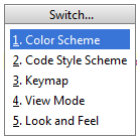
1. In the Settings/Preferences dialog , expand the Tools node and choose Settings Repository .
2. Click + and add the URL of the GitHub repository that contains the settings you want to share.

Synchronization with the settings from read-only sources is performed in the same way as for the Settings Repository .

Switching Between Schemes

You can quickly switch between various color schemes, keyboard layouts, and look-and-feels without actually invoking the corresponding page of the Settings dialog box.

1. Choose View | Quick Switch Scheme on the main menu or press `Ctrl+Back Quote`.
2. In the pop-up window that opens select the desired scheme (Colors and Fonts, Code Style, etc.).



3. In the suggestion list, click the desired option.

Switching Boot JDK

Warning! Regardless of your choice, the selected version of JDK shall be not lower than 1.8.
– Currently, this option is available for OS X and Linux users only

On this page:

– Introduction
– Switching the IDE boot JDK

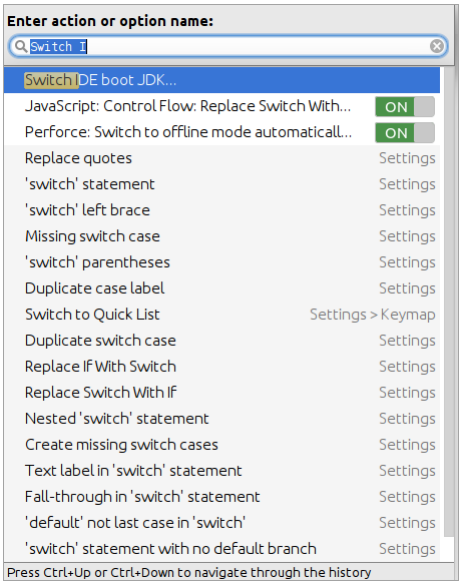
Introduction

In case when you prefer JDK other than bundled with PhpStorm, you can choose between the latter and another kit, installed on your system.

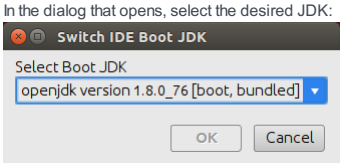
Switching the IDE boot JDK

To switch the IDE boot JDK, do the following:

1. On the main menu, choose Help | Find Action or press `Ctrl+Shift+A`.
2. In the list of actions that appears, find Switch IDE boot JDK action and select it. Simplify your search by typing the first letters:



3. In the dialog that opens, select the desired JDK:



4. Click OK to apply changes.

Configuring Project Structure

In this part:

- Configuring Project Structure
  - Basics
- Configuring Content Roots
- Configuring Folders Within a Content Root
- Excluding Files from Project

Basics

To access project structure, use the Directories page of the Settings/Preferences dialog, which is invoked by icon on the main toolbar, or keyboard shortcut `Ctrl+Alt+S`.

Configuring Content Roots

On this page:

- Introduction
- Creating a content root
- Removing a content root

Introduction

Any project contains at least one content root created together with the project.

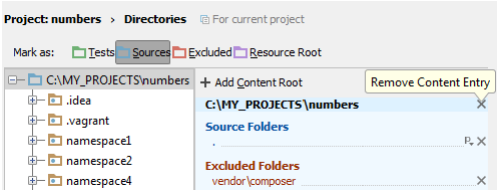
You can create additional content roots as well as remove the unnecessary ones.

Creating a content root

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Directories under Project:<project name>.
2. On the Directories page, click the Add Content Root button +.
3. In the dialog that opens , locate the desired directory and click OK.

Removing a content root

- Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Directories under Project:<project name>.
- On the Directories page, select the content root to be deleted.
- Click the Remove button .



- Confirm deletion.

Configuring Folders Within a Content Root

In this section:

- Overview
- Marking directories
- Unmarking directories

Overview

Within a content root , PhpStorm can distinguish between the folders that contain source code, and the ones to be ignored while searching, parsing, watching etc. To do so, you can mark any folder below a content root as a source folder, or as excluded so it becomes invisible for PhpStorm.

Within a content root, PhpStorm can distinguish between Source folders, Resource Root folders, Test Source code, and Excluded folders.

The folder marked as Source is hte root folder for all the PHP namespaces in the current project.

Files under a folder marked as Resource Root can be referenced relative to this folder. Excluded folders are not involved in indexing and therefore they are ignored while searching, parsing, watching etc.

To invoke this distinction, you can mark any folder below a content root as Source, Resource Root, Test, or Excluded.

Marking directories

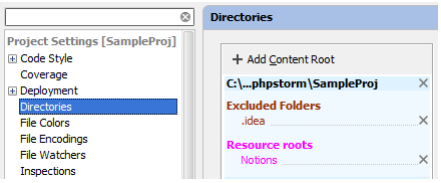
You can assign a folder to a category in two different ways:

- Using the Directories page of the Settings / Preferences Dialog .
- Using the context menu of a folder in the Project tool window .

To mark directories under the content root:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Click the Directories node.
2. In the Directories page, click the desired content root. The directories under this content root are displayed as a tree view in the right-hand pane.
3. Select the directory you want to mark and do one of the following:
   - To have PhpStorm consider the contents of the selected folder as unit tests, click the Tests toolbar button or choose Test Sources on the context menu of the selection.
   - To have PhpStorm consider the selected folder as the root for namespaces used in your project, click the Sources toolbar button o choose Sources on the context menu of the selection.
     Based on this setting, PhpStorm suggests you the proper folder name when you want to create a new namespace under another parent namespace during creation or moving a PHP class, that is, when you are actually creating or moving a PHP class to a non-existing namespace under another parent namespace. If no Sources folder is specified, you will have to type the proper folder manually.

     Appointing a Sources folder is not mandatory but this helps you keep your project structure in compliance with the PSR0 and PSR4 standards. See Configuring PHP Namespaces in a Project for details.
   - To have PhpStorm ignore the selected directory during indexing, parsing, code completion, etc., click the Excluded toolbar button or choose Excluded on the context menu of the selection.
   - To enable PhpStorm to complete relative paths to resources under the selected folder, click the Resource Root toolbar button or choose Resource Root on the context menu of the selection.



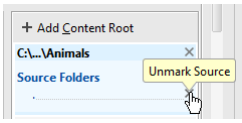
To mark directories using the context menu:

1. Right-click the desired directory in the Project Tool Window .
2. On the context menu, point to Mark Directory As node.
3. Choose Mark as <directory status>.

Unmarking directories

To return a folder to its regular status, do one of the following

- On the Directories page of the Settings / Preferences Dialog :
  - Select the directory in question in the list of folders under the content root, and click :



  - Click the folder's status icon once more.
  - Choose the corresponding command on the context menu of the directory.
- In the Project Tool Window , right-click the desired directory, point to Mark Directory As node, and then choose Unmark as <directory status>.

>

Excluding Files from Project

On this page:

- Basics
- Marking a file as plain text
- Marking a file with its regular type
- Excluding a .phar archive from a project

Basics

Sometimes you might need to exclude a single file from your project, so that it will be ignored by inspections, code completion, etc. This is done using the Mark as plain text action.

When a file is marked as plain text, PhpStorm does not use it anymore for code completion and navigation. Such file is shown as plain text in the editor, and is denoted with a special icon in the Project tool window .

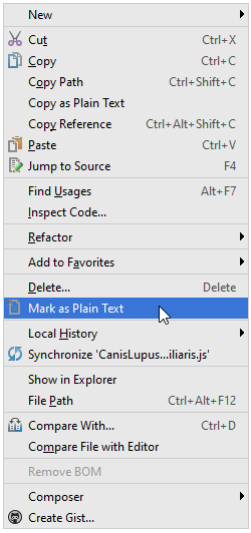
The reverse action is also available: you can return a file to its original type, using the Mark as <file type> action.

In the context of a PHP project, you can also exclude entire PHP archives (`.phar` files) to prevent multiple declaration conflicts between classes in a `.phar` archive and the corresponding classes in the installed vendors directory.

Marking a file as plain text

To mark a file as plain text, follow these steps

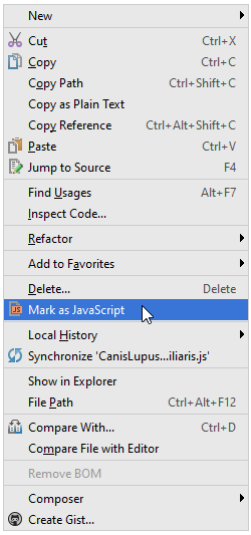
1. In the Project tool window , select the desired file.
2. On the context menu of the selection, choose Mark as plain text:



Marking a file with its regular type

To mark a file with its regular type, follow these steps

1. In the Project tool window , select the desired file, marked with icon.
2. On the context menu of the selection, choose Mark as <file type>:



Excluding a .phar archive from a project

To exclude a .phar archive from a project

- In the Project tool window, select the required `.phar` archive and choose Exclude phar from project on the context menu of the selection.

Creating and Managing Projects

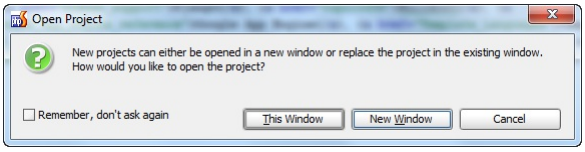
In this section:

- Creating New Project from Scratch
- Importing Project from Existing Source Code
- Generating a Project from a Framework Template
- Creating a Project Using Yeoman Generator
- Opening, Reopening and Closing Projects
- Opening Multiple Projects
- Saving Projects as Templates
- Cleaning System Cache
- Renaming Projects

Creating New Project from Scratch

1. Do one of the following:
   - On the Welcome screen , under the Quick Start section, click the link Create New Project.
   - On the main menu, choose File | New | Project.
2. In the Select Project Type dialog box that opens, choose PHP Empty Project in the left-hand pane.
3. In the Location text box, specify the project name and location.
4. In the PHP Language Level drop-down list, specify the PHP functionality scope to get coding assistance for. Each functionality scope is associated with PHP version that supports this functionality. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported.
   No correlation between the PHP version used in the project and the language level is enforced. Although the language version of each interpreter is detected automatically, you can still tell PhpStorm to provide you with coding assistance that corresponds to another language level. However, if you attempt to use a code construct that is not supported by the specified language level, PhpStorm suggests a Switch to PHP <version> quick-fix .
5. From the Interpreter drop-down list, choose the PHP interpreter to use in the current project. The list contains all the currently configured local and remote PHP interpreters. See Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters for details.
6. The Include Path area shows a list of paths to PHP-related items below the PHP home directory. The specified include paths will be used:
   - By the `require()` , `include()` , `fopen()` , `file()` , `readfile()` , and `file_get_contents()` functions when looking for files to use.
   - By PhpStorm when resolving references to included files.

   Use the Add + and Remove — buttons to manage the contents of the list. Use the Up ↑ and Down ↓ buttons to change the order of items in the list.
7. When you click Create, PhpStorm generates a project stub and opens it either in the current window or in a new one depending on your choice in the information dialog box that opens:



Refer to the section Opening Multiple Projects for details.

Importing Project from Existing Source Code

You can set up a project around the existing source code created externally, in other words, in another IDE or in a dedicated editor. PhpStorm analyzes the code base, adds the `.idea` directory with settings, and marks the project with the special icon .

When the files to work with are already on your machine, you can set up your development environment in three main ways:

- You have a Web server on your machine, the sources are under the server document root (for example, `/htdocs`), and you do your development right on the server.
- The Web server is installed on your computer but the sources are stored in another folder or on a mounted drive. You do your development, then synchronize with the server.
- You do not have a Web server on your machine or you are not going to use it right now. The sources are on your machine, you just do your development, possibly using the console to view the output.

Finally, you may need to update an application on a remote host. In this case, you also have to download the application sources and arrange them in a project, because PhpStorm does not support direct editing files on remote hosts. PhpStorm assumes that all development, debugging, and testing is done on your computer and then the code is deployed to a production environment. For more details, see the sections Working with Web Servers and Creating a Project from Downloaded Files .

In this section:

- Creating a Project from Existing Local Sources
- Creating a Project from Downloaded Files

Creating a Project from Existing Local Sources

When the files to work with are already on your machine, you can set up your development environment in three main ways:

- You do not have a Web server on your machine or you are not going to use it right now. The sources are on your machine, you just do your development, possibly using the console to view the output. In this case, you can just open the folder under which the sources are stored without configuring access to a server. However, you can also use the New Project wizard.
- You have a Web server on your machine, the sources are under the server document root (for example, `/htdocs`), and you do your development right on the server. In this case, you need to use the New Project wizard to arrange the sources in a project, appoint the project root folder, and configure access to the server through a server access configuration of the type in-place.
- The Web server is installed on your computer but the sources are stored in another folder or on a mounted drive. You do your development, then synchronize with the server. Use the New Project wizard to arrange the sources in a project, appoint the project root folder, configure access to the server through a server access configuration of the type local, and map the folders under the project root folder with folders on the server.

On this page:

- Creating a Project by Opening the Folder where the Sources Are Stored
- Creating a Project Around Existing Sources Without a Server Using the New Project Wizard
- Creating a Project from Existing Sources Under the Server Document Root
- Creating a Project from Sources Outside the Server Document Root
- Configuring folders under the project root

Creating a Project by Opening the Folder where the Sources Are Stored

If you are not going to use a Web server right now, you can just open the folder with the files to be arranged in a PhpStorm project.

1. On the main menu, choose File | Open.
2. In the dialog that opens , select the directory that contains the desired source code. Note that applications created externally are marked with the regular directory icon.
3. Click OK and specify whether you want the new project to be opened in a separate window or close the current project and reuse the existing one. Refer to the section Opening Multiple Projects for details.

Creating a Project Around Existing Sources Without a Server Using the New Project Wizard

Another way to arrange the source in a PhpStorm project is use the New Project wizard.

1. To start the New Project wizard, choose New Project from Existing Files on the main menu.
2. On the Choose Your Scenario page , choose Source files are in a local directory, no Web server is yet configured and click Next.
3. On the Choose Project Directory page , select the folder where your sources are stored and click the icon on the toolbar or choose Project Root on the context menu of the selection to appoint the folder project root . PhpStorm marks the selected folder with the icon.
4. Configure the folders under the project root . Alternatively, you can do it later on the Directories page of the Settings dialog box.
5. Click Finish and specify whether you want the new project to be opened in a separate window or close the current project and reuse the existing one. Refer to the section Opening Multiple Projects for details.

Creating a Project from Existing Sources Under the Server Document Root

In this case, you need to use the New Project wizard to arrange the sources in a project, appoint the project root folder, and configure access to the server through a server access configuration of the type in-place. In this case, you need to use the New Project wizard to arrange the sources in a project, appoint the project root folder, and configure access to the server.

1. To start the New Project wizard, choose New Project from Existing Files on the main menu.
2. On the Choose Your Scenario page , choose Web server is installed locally, source files are located under its document root and click Next.
3. On the Choose Project Directory page , select the folder where your sources are stored and click the icon on the toolbar or choose Project Root on the context menu of the selection to appoint the folder project root . PhpStorm marks the selected folder with the icon.
4. Configure the folders under the project root . Alternatively, you can do it later on the Directories page of the Settings dialog box. Click Next.
5. On the Specify Local Server page , appoint the server access configuration to use. For more details, see Configuring Synchronization with a Web Server .
   Do one of the following:
   - Choose Use existing server and select the relevant server configurations from the list. The list contains only configurations of the type In-place marked with , because your files are already under the server document root.
   - Choose Add new local server, click Next, and define a new server access configuration on the Add Local Server page that opens:
     1. Specify the server configuration name in the Name text box. PhpStorm itself sets the configuration type to In-place.
     2. In the Web server root URL text box, type the URL address associated with the document root of your Web server as defined in the Web server configuration file.
        This URL address will be the starting point for building the URL address of your application. Both the HTTP and the HTTPS protocols are supported.
        To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`.
   Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

For example, the Apache httpd server configuration file is `httpd.conf` . The default document root is the `htdocs` folder and the default URL address to access the data in it is `http://localhost` . If you have changed the default port `80` , you have to specify the port explicitly: `http://localhost:<port>` .

Click Next, when ready.

3. On the Specify Web Path page , specify the URL address to access your project root folder. In the Web path for project root text box, type the path to the project root folder relative to the server document root specified in the server configuration file. As you type, PhpStorm dynamically builds the URL address and shows as a link in the Project URL read-only field. To check that the URL address is constructed correctly and ensures access to the project root, click the link.

For more details, see Creating an in-Place Server Configuration .

6. Click Finish and specify whether you want the new project to be opened in a separate window or close the current project and reuse the existing one. Refer to the section Opening Multiple Projects for details.

Creating a Project from Sources Outside the Server Document Root

In this case, you need to use the New Project wizard to arrange the sources in a project, appoint the project root folder, and configure access to the server.

1. To start the New Project wizard, choose New Project from Existing Files on the main menu.
2. On the Choose Your Scenario page , choose Web server is installed locally, source files are located elsewhere locally and click Next.
3. On the Specify Local Path page , appoint the project root. Specify the folder where the sources are stored and where the project will be created.
4. On the Specify Server page, appoint the server access configuration to enable interaction with the server. For more details, see Configuring Synchronization with a Web Server .
   Do one of the following:
   - Choose Use existing server and select the relevant server configurations from the list. The list contains only configurations of the type Local marked the , because your files are stored locally on your machine but under the server document root, so a configuration fo the type In-place would not fit.
   - Choose Add new local server, click Next, and define a new server access configuration on the Add Server page, that opens:
     1. Specify the server configuration name in the Name text box. PhpStorm itself sets the configuration type to Local.
     2. In the Folder text box of the Upload/download project files area, specify the server configuration root.
        The server configuration root is the highest folder in the file tree on the server that can be accessed through the server configuration. The easiest way is to use the document root of your Web server as defined in the Web server configuration file. However you can appoint any other existing folder under the document root.
     3. In the Web server root URL text box of the Browse files on server area, Click Next.

   For more details and examples, see Creating a Local Server Configuration .
5. On the Choose Remote Path page, create a folder on the server where PhpStorm will upload the project files. This folder is called project root on the server and it will be automatically mapped to your project root.
   The page shows a tree of the folders under the appointed server configuration root. Select the relevant parent folder and choose Create Folder on the context menu. When the new folder appears in the tree, select it and click the icon on the toolbar or choose Project Root on the context menu of the selection. PhpStorm marks the selected folder with the icon.

   Configure the folders under the project root . Alternatively, you can do it later on the Directories page of the Settings dialog box. Click Next.
6. On the Specify Web Path page , specify the URL address to access your project root folder on the server which you appointed in the previous step. In the Web path for project root text box, type the path to the folder relative to the server configuration root. Actually, type the relative path to the project root on the server text box. As you type, PhpStorm dynamically builds the URL address and shows it as a link in the Project URL read-only field.

Configuring folders under the project root

PhpStorm can distinguish between Source folders, Resource Root folders, and Excluded folders. By default, any folder is treated as Source folder unless it is associated with another type.

You may also need to suppress upload/download of files and folders. Suppressing uploading, downloading, and synchronization for files or folders with sources ensures that the sources are protected against accidental update. When applied to non-sources, it saves system resources because huge amounts of media, caches, or temporal files are no longer copied hither and thither.

Select the directory you want to mark and do one of the following:

- To have PhpStorm ignore the selected directory during indexing, parsing, code completion, etc., click the Excluded from Indexing toolbar button or choose Excluded on the context menu of the selection.
- To enable PhpStorm to complete relative paths to resources under the selected folder, click the Resource Root toolbar button or choose Resource Root on the context menu of the selection.
- To suppress uploading, downloading, and synchronization for the selected folder, click the Excluded from Download icon or choose Exclude Path on the context menu of the selection.

For details, see Configuring folders under content roots and Excluding Files and Folders from Upload/Download .

Creating a Project from Downloaded Files

From time to time, you may need to update externally created and uploaded to a remote host with access through the FTP/SFTP/FTPS protocol, network share, or mounted drive. PhpStorm does not support direct editing of files on remote hosts. Therefore to work with remote sources in PhpStorm, you need to download them and arrange them in a PhpStorm project . To keep your local and remote or shared sources synchronized, you can configure automatic upload using the Upload changed files automatically to the default server drop-down list in the Options dialog box.

PhpStorm interacts with the remote host where the files are located according to a server access configuration. You can use an appropriate existing configuration or define a new one during project creation, see Working with Web Servers: Copying Files for details.

Generating a Project from a Framework Template

On this page:

- Overview
- Generating a HTML5 Boilerplate, Web Starter Kit, React Starter Kit, Twitter Bootstrap, or Foundation application stub
- Generating a Node.js Express application stub
- Generating a Dart application stub
- Generating a Meteor application stub
- Generating a PhoneGap/Cordova/Ionic application stub
- Generating an AngularJS application stub
- Generating a Composer project stub
- Generating a WordPress application stub
- Generating a Drupal module stub
- Generating a Joomla! project stub

Overview

During project creation, PhpStorm can generate a project stub for developing Web applications. The project structure is set up and some sources are generated based on external templates and frameworks downloaded upon your request.

PhpStorm generates project stubs based on the following templates:

- Composer project.
- Drupal module.
- WordPress project.
- HTML5 Boilerplate , Twitter Bootstrap , and Foundation for client-side application stubs.
- Node.js Express for the server-side applications using Node.js . and Node.js Boilerplate
- Dart Web application stubs for developing the client side of applications.
- Meteor project stubs for both the client and the server sides of applications.
- PhoneGap / Apache Cordova / Ionic project stubs.
- AngularJS project stubs.

Generating a HTML5 Boilerplate, Web Starter Kit, React Starter Kit, Twitter Bootstrap, or Foundation application stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose the template to use depending on your preferences and the functionality to be implemented in your application:
   - Empty: choose this option to get just a project folder without any contents.
   - Composer Project: choose this option to have a project stub created using the Composer template .
   - Drupal Module: choose this option to have a Drupal module created and set up.
   - HTML5 Boilerplate: choose this option to use the HTML5 Boilerplate template. This is a starting project template that can be easily adapted to your needs.
   - Web Starter Kit: choose this option to use the Web Starter Kit .
   - React Starter Kit: choose this option to use the React Starter Kit .
   - Twitter Bootstrap: choose this option to use the Twitter Bootstrap template, which is an elaborate modular toolkit with rich HTML, CSS, and JavaScript support.
   - Foundation: choose this option to use the Foundation framework.
3. The set of controls in the right-hand pane depends on the chosen template.
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Choose the template version to use and click Create.

Generating a Node.js Express application stub

Generating such application stubs requires that Node.js is supported in PhpStorm:

1. The Node.js runtime environment is downloaded and installed on your computer.
2. The Node.js repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Learn more in Node.js

To generate a Node.js application stub:

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Node.js Express App.
3. In the right-hand pane, specify the following:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. The path to the Node.js executable file `node.exe` and to the Node.js package manager file `npm.cmd` .
   3. The Express template engine to use. From the Template engine drop-down list, choose one of the following:
      - Jade - haml.js successor.
      - EJS - embedded JavaScript.
      - Hogan.js .
      - Handlebars .
   4. The CSS engine to use. From the CSS engine drop-down list, choose one of the following:
      - Plain CSS
      - Stylus
      - Less

- Compass .
- Sass .

4. Click Create, when ready. PhpStorm launches the Express Project Generator tool that downloads all the required data, sets up the project structure, and opens the project either in the current window or in a new one, depending on your choice.

Generating a Dart application stub

Generating Dart application stubs requires that Dart is supported in PhpStorm:

1. The Dart SDK is available on your computer.
2. The Dart plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

To generate a Dart application stub:

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Dart.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the paths to the Dart SDK and to the Dartium executable file. PhpStorm detects and displays the Dart version. If the SDK was downloaded within the Dart Editor Bundle, the path to Dartium is also detected and filled in automatically.
   3. To have annotations verified and assertions checked, select the Start Dartium in checked mode check box. Learn more about Dartium command line flags at https://www.dartlang.org/tools/dartium/ .

   Click Create when ready.

If necessary, you can open the Dart page of the Settings dialog box by clicking the Settings button and configure command-line options and a custom user data directory for Dartium.

Generating a Meteor application stub

When you click Create, PhpStorm generates a skeleton of a Meteor application, including an HTML file, a JavaScript file, a CSS file, and a `.meteor` folder with subfolders. The `.meteor/local` folder, which is intended for storing the built application, is automatically marked as excluded and is not involved in indexing.

By default, excluded files are shown in the project tree. To have the `.meteor/local` folder hidden, click the button on the toolbar of the Project tool window and remove a tick next to the Show Excluded Files option.

PhpStorm also automatically attaches the predefined Meteor library to the project, thus enabling syntax highlighting, resolving references, and code completion. See Configuring JavaScript Libraries for details.

Meteor uses Spacebars templates that are an extension of the Handlebars/Mustache templates. PhpStorm recognizes Spacebars templates, but as a side effect marks HTML files in Meteor projects with the Handlebars/Mustache icon .

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Meteor JS App.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the location of the Meteor executable file (see Installing Meteor).
   3. From the Template drop-down list, choose the sample to generate. To have a basic project structure generated, choose the Default option.
   4. In the Filename text box, type the name for the mutually related `.js` , `.html` , and `.css` files that will be generated. The text box is available only if the Default sample type is selected from the Template drop-dow list.

Generating a PhoneGap/Cordova/Ionic application stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose PhoneGap/Cordova App.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the location of the executable file `phonegap.cmd` , or `cordova.cmd` , or `ionic.cmd` (see Installing PhoneGap/Cordova/Ionic).

When you click Create, PhpStorm generates a skeleton of a PhoneGap/Cordova/Ionic application with the framework-specific structure.

Generating an AngularJS application stub

This requires that the AngularJS plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

To generate an AngularJS application stub:

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose AngularJS.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. From the Version drop-down list, choose the branch https://github.com/angular/angular.js to download the project template from. By default, `master` if shown.
4. When you click Create, PhpStorm sets up the AngularJS-specific project structure and generates the `bower.json` and `package.json` files with the definitions of all the necessary dependencies.
5. Download the AngularJs dependencies that contain the AngularJS code and the tools that support development and testing.
   1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).

2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
3. At the command line prompt, type the following command:

```
npm install
```

Learn more about the installation of dependencies in the Install Dependencies section of the `readme.md` file.

When you click Create, PhpStorm generates a skeleton of a AngularJS application with the framework-specific structure.

Generating a Composer project stub

You can have PhpStorm generate a project stub from an existing package using Composer. In this case, Composer will be initialized and one package will be installed during the project stub creation. After that you can use Composer from the command line or through the user interface.

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Composer Project.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Composer-specific information:
   1. Appoint the Composer instance to use.
      - To use commands from a previously downloaded `composer.phar` , choose Use existing composer.phar: and specify it location in the text box.
      - To have a new instance of Composer downloaded, choose Download composer.phar from getcomposer.org:. The `composer.phar` file will be saved under the project root folder specified in the Location text box of the New Project Dialog dialog.
   2. In the Package area, specify the package to install during the project creation. Select the package to add from the Available Packages list, possibly using the search field, and choose the relevant version from the Version to install drop-down list.
   3. Expand the Settings area and specify the path to the PHP interpreter to use (see Configuring Remote PHP Interpreters and Configuring Local PHP Interpreters for details). In the Command line parameters text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have a package added to the `require-dev` section instead of the default `require` section type `-dev` . For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .

When you click Create, the `create-project` command is invoked with the selected package. This results in creating a Composer project whose configuration and structure depends on the selected package, see https://getcomposer.org/doc/03-cli.md#create-project for details. After that a PhpStorm project opens.

Generating a WordPress application stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose WordPress Plugin.
3. In the right-hand pane, specify the folder where the project will be created. In the WordPress Installation Path text box, specify the folder where WordPress is installed. This folder should contain the `wp-admin` and `wp-includes` subdirectories. Click Create to start the project stub generation.
4. If the newly created project is outside the WordPress installation folder, configure it as an external library by adding it to the list of included path.
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, add the path to the WordPress installation folder to the Include Paths list: click the + button and specify the path to the installation folder in the dialog box that opens. Learn more in Configuring Include Paths .

Generating a Drupal module stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Drupal Module.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Drupal-specific information:
   1. Specify the root folder of the Drupal installation in the Drupal installation path. Type the path manually or click the Browse button ... and select the relevant folder in the dialog box that opens.
   2. Select the Set up PHP | Include paths check box to have Drupal include paths automatically configured for the project. The following paths will be added to the Include Paths list on the PHP page of the Settings dialog box:
      1. `<drupal installation root>/includes`
      2. `<drupal installation root>/modules`
      3. `<drupal installation root>/sites/all/modules`

      The include paths are added to the list after you leave the dialog box.
   3. From the Version drop-down list, choose the version of Drupal to use, the supported versions are 6, 7, and 8 .

You can update integration settings at any time later in the File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.

PhpStorm generates a project stub in accordance with the selected Drupal version. For Drupal 8, a `module_name.info.yml` file is generated where information about the module is provided.

Generating a Joomla! project stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Joomla! Integration.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Joomla!-specific information:
   - Joomla installation path
   - Project type (component, module or plugin)

   PhpStorm shows balloons with notifications about the missing Joomla! code styles, namespaces etc. These elements can be installed straight away by clicking the corresponding links. The notifications are duplicated in the Status Bar , and in the Events tool window .

PhpStorm creates a stub Joomla! project that contains a PHP class `<project name>.php` and `<project name>.xml` manifest file.

Refer to the section Using Joomla! with PhpStorm and to Joomla! site.

Creating a Project Using Yeoman Generator

PhpStorm supports integration with the Yeoman tool and provides interface for using it in generating framework-specific project stubs.

On this page:

- Before you start
- Installing Yeoman
- Configuring a list of project stub generators
- Creating a project by a generator

Before you start

1. Download and install Node.js. The runtime environment is required for two reasons:
   - The Yeoman tool is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Yeoman tool.
2. If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Yeoman tool and npm from any folder.
3. Make sure the Yeoman and the NodeJS plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Installing Yeoman

The easiest way to install Yeoman is to use the Node Package Manager (npm), which is a part of Node.js. Yeoman can be installed both globally or locally, in a project. It is recommended that you install globally because in this case you can run it from any folder but not from its installation folder.

PhpStorm provides interface for global and local installation modes. Alternatively, you can install Yeoman manually through the command line.

To install Yeoman from PhpStorm using the Node.js and NPM page of the Settings dialog box:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the yo package from the list.
4. For global installation, select the Options check box and type `-g` in the text box next to it.
5. Click Install Package to start installation.

To run the installation from the command line:

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Type the following command at the command line prompt:

```
npm install -g yo
```

   where `-g` means global installation

Configuring a list of project stub generators

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Yeoman.
3. The right-hand pane shows all the previously installed Yeoman generators . Click Install Generator.
4. From the dialog box that opens showing all the available generator packages, select the required package in the left-hand pane and click the Install Generator button that appears in the right-hand pane. You can install several packages one after another without leaving the dialog box.
   When the installation is over, click Close to return to the list of generators which is already expanded with the newly added package.

Creating a project by a generator

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Yeoman.
3. The right-hand pane shows all the previously installed Yeoman generators . To expand the list, click Install Generator, see Creating a Project Using Yeoman Generator above.
   Select the required generator from the list and click Next.
4. Specify the required settings in the New Project wizard that starts. The number of pages and their contents depend on the chosen generator.
5. On the last page of the wizard, select or clear the Run npm install&bower install check box to specify whether you want to run Node Package Manager and Bower to install the packages that are required for developing the new project.
6. Click Next and choose to open the new project in the current window or in the new one.

Opening, Reopening and Closing Projects

On this page:

- Basics
- Opening an existing project
- Reopening a recent project
- Closing a project

Basics

PhpStorm allows opening several projects simultaneously in different windows. By default, each time you open a project while another one is opened, PhpStorm prompts you to choose whether to open the project in the same window or in a new window. If necessary, you can change this behavior using controls on the System Settings page.

In the Project Opening section of this page, you can choose one of the following options:

- Open project in a new window to open a new PhpStorm window each time a new project is opened.
- Open project in the same window to stay in the same PhpStorm window.
- Confirm window to open project in to keep the default behavior and display a dialog box to choose how to open each new project.

Opening an existing project

To open an existing project

1. Do one of the following:
   - On the Welcome screen , click Open.
   - On the main menu, choose File | Open.
2. In the Select Path dialog box, select the directory that contains the desired project.

   Tip Drag the desired project from your file chooser right to the Open Project dialog without locating it there. The respective file in the dialog will be found automatically.
3. Specify whether you want to open the project in a new frame, close the current project and reuse the existing frame, or open the new project in the same frame with the current one. Refer to the section Opening Multiple Projects for details.

Reopening a recent project

PhpStorm keeps the history list of the recent projects, from which you can select the desired one.

When PhpStorm starts, the most recent project reopens by default (unless this option is disabled in System Settings page of the Settings/Preferences dialog).

To reopen a recent project

1. On the main menu, choose File | Open Recent.
2. Select the desired project from the list of the recent ones.
3. Specify whether you want to open the project in a new frame, or reuse the current frame.

You can also:

- Reopen a recent project from the Welcome screen . To do that, click the project of interest in the left-hand pane where the list of your recent projects is shown. Start typing in the Welcome screen to filter the list of recent projects.
- Terminate the project opening process, without waiting when it finishes loading, by clicking Cancel in the progress window.
- Open projects and individual files from the command line, by specifying the project or file name as a command line argument.
- Remove all projects from the list of recent projects File | Open Recent - Clear List.
- Delete a certain recent project on the Welcome screen.

Closing a project

When the only open project is closed, the Welcome screen is displayed. In case of multiple projects, each one is closed with its frame.

To close a project

- On the main menu, choose File | Close Project.

Opening Multiple Projects

On this page:

- Basics
- Prerequisite
- Opening project in a new window

Basics

PhpStorm allows you to work with several projects simultaneously, each one in a separate window. The projects are independent, and cannot share information, except for the Clipboard operations. All the projects run in the same instance of PhpStorm and use the same memory space.

Tip This behavior is controlled by the options in the section Project opening in the System Settings page of the Settings dialog.

Prerequisite

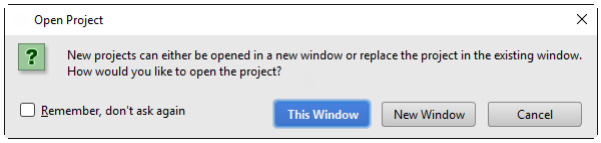
Make sure that one of the options Open project in a new window or Confirm window to open project in in the System Settings page of the Settings/Preferences dialog is selected.

Opening project in a new window

To open a project in a new window, follow these steps

1. Open a project , while another one is already open.

2. Depending on the selected behavior:
   - If the option Open project in a new window is selected, PhpStorm will silently open new project in a new window.
   - If the option Confirm window to open project in is selected, PhpStorm will prompt you to specify whether you want to open project in a new window, or close the current project and reuse the existing window.



   Click the New Window button to leave the current project in its window and open the new one in a separate window.

   Click This Window button to close the current project and opening the new one in the same window.

Tip PhpStorm suggests to memorize your choice by selecting the check box Remember, don't ask again .

Saving Projects as Templates

On this page:

- Introduction
- Creating a project template
- Deleting a project template

Introduction

You can save any project as a template. If you do so, you'll be able to create template-based projects, that is, projects that are nearly identical to the template project.

The projects that you have saved as templates appear in the User-defined category in the New Project Dialog dialog.

Unnecessary templates can be deleted in the Manage Project Templates Dialog .

Creating a project template

1. Open the desired project in PhpStorm.
2. In the main menu, select Tools | Save Project as Template.
3. In the Save Project as Template Dialog that opens:
   - Specify the template name and optionally provide a brief template description.

Deleting a project template

1. In the main menu, select Tools | Manage Project Templates.
2. In the Manage Project Templates Dialog that opens, select the template or templates you want to delete.
3. Click — or press `Alt+Delete`.

Cleaning System Cache

PhpStorm caches a great number of files, therefore the system cache may one day become overloaded. In certain situations the caches will never be needed again, for example, if you work with frequent short-term projects. Also, the only way to solve some conflicts is to clean out the cache.

Warning! Cleaning out the system caches, keep in mind that:

- It results in clearing the local history.

  To avoid losing data, check in the changes to your version control system before invalidating caches.
- Causes a complete rebuild of all the projects ever run in the current version of PhpStorm.

Cleaning out the system caches

To clean out the system caches:

- On the main menu, choose File | Invalidate Caches/Restart. The Invalidate Caches message appears informing you that the caches will be invalidated and rebuilt on the next start. Use buttons in the dialog to invalidate caches, restart PhpStorm or both.

It is important to note the following:

- The files are not actually deleted until PhpStorm restarts.
- Opening and closing a project does not result in deleting any files.

Renaming Projects

Normally, a project in PhpStorm has the same name as its root folder. So the easiest way to change the name of a project is just rename its root folder.

However in this case you have to re-configure access to the server so the server root is mapped to the new project root name. To avoid these additional steps, you can change the name of the project but have its root folder name preserved.

To rename a project without changing the name of its root folder

1. On the main menu, choose File | Rename Project.
2. In the dialog box that opens, type the new name of the project.

File and Code Templates

In this part:

- File and Code Templates
  - Overview
  - Per-project vs default scheme
  - Predefined, internal, and custom templates
  - When are file and code templates used?
- #Parse Directive
- File Template Variables
- Creating and Editing File Templates

Overview

File templates are specifications of the contents to be generated when creating a new file. They let you create the source files that already contain some initial code.

You can view, edit and create the templates on the File and Code Templates page of the Settings/ Preferences dialog.

File and code templates are written in the Velocity Template Language (VTL).

So they may include:

- Fixed text (markup, code, comments, etc.). In a file based on a template, the fixed text is used literally, as-is.
- File template variables . When creating a file, the variables are replaced with their values.
- #parse directives to include other templates defined in the Includes tab on the File and Code Templates page of the Settings/Preferences dialog box.
- Other VTL constructs.

Here is a typical template example. (This template is used for creating a JavaScript file.)

```
/**
                    * Created by ${USER} on ${DATE}
                    */
```

In this template, `${USER}` and `${DATE}` are template variables.

Applying this template leads to generating a file whose contents look similar to this:

```
/**
                    * Created by John.Smith on 6/7/11
                    */
```

Per-project vs default scheme

PhpStorm suggests using file and code templates on the project or default (global) level.

If you need a sharable set of file and code templates, then these templates should be per-project; otherwise the templates are global and pertain to the entire workspace.

The file and code templates are stored in the following locations:

- The default (global) templates are stored in the PhpStorm home directory, in the folder `config | fileTemplates`.
- The per-project file and code templates are stored in the `.idea | fileTemplates` folder. These templates can be shared among the team members.

Refer to the section Project and IDE Settings to learn where the settings are stored, and to the File and Code Templates dialog for the description of the Schema field.

Predefined, internal, and custom templates

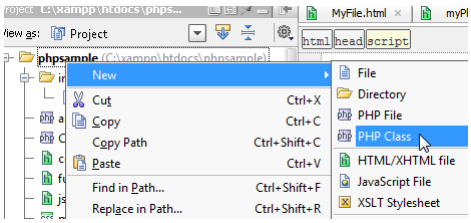
PhpStorm comes with a set of predefined file and code templates. You can use these templates as-is or modify them as necessary. You can as well create your own templates (custom templates).

Internal file and code templates are a subset of the predefined templates. These templates differ from all the other templates in that they cannot be deleted.

On the File and Code Templates page of the Settings/Preferences dialog, the names of internal templates are shown in bold. The names of the custom templates and the predefined templates that you have modified are shown in blue.

When are file and code templates used?

Whenever you create a new file, you can choose to create an empty file (e.g. File | New | File) or use a file template. In the latter case, the initial contents of the new file will be generated according to the template you have selected. (Basically, all the options in the New menu except File and Directory correspond to using a template.)



#parse Directive

Using the `#parse` directive, you can include other templates in file templates . This is useful for inserting reusable contents (e.g. standard headers, copyright statements, etc.) into multiple file templates.

The syntax for the `#parse` directive is:

```
#parse("<template_name.extension>")
```

For example: `#parse("ActionScript File Header.as")`.

The templates that can be referenced like this in other templates, are shown on the Includes tab of the File and Code Templates settings page.

File Template Variables

On this page:

- Basics
- Predefined template variables
- Custom template variables

Basics

A file template can contain template variables. When a template is applied, the variables are replaced with their values.

A file template variable is a string that starts with a dollar sign which is followed by the variable name. The variable name may be enclosed in curly braces. For example: `$MyVariable` or `${MyVariable}`.

Predefined template variables

PhpStorm comes with a set of predefined template variables.

The available predefined file template variables are:

- `${PROJECT_NAME}` - the name of the current project.
- `${FILE_NAME}` - the name of the PHP file that will be created.
- `${NAME}` - the name of the new file which you specify in the New File dialog box during the file creation.
- `${USER}` - the login name of the current user.
- `${DATE}` - the current system date.
- `${TIME}` - the current system time.
- `${YEAR}` - the current year.
- `${MONTH}` - the current month.
- `${DAY}` - the current day of the month.
- `${HOUR}` - the current hour.
- `${MINUTE}` - the current minute.
- `${PRODUCT_NAME}` - the name of the IDE in which the file will be created.
- `${MONTH_NAME_SHORT}` - the first 3 letters of the month name. Example: Jan, Feb, etc.
- `${MONTH_NAME_FULL}` - full name of a month. Example: January, February, etc.

PhpStorm provides a set of additional variables for PHP include templates , that is, templates of reusable fragments that can be included in other PHP file template. The built-in PHP include templates are intended for generating file headers and PHPDoc documentation comments . The following variables are available in PHP include templates:

- `${NAME}` - the name of the class, field, or function (method) for which the PHPDoc comment will be generated.
- `${NAMESPACE}` - the fully qualified name (without a leading slash) of the class or field namespace.
- `${CLASS_NAME}` - the name of the class where the field to generate the PHPDoc comment for is defined.
- `${STATIC}` - gets the value `static` if the function (method) or field to generate the comment for is `static`. Otherwise evaluates to an `empty string`.
- `${TYPE_HINT}` - a prompt for the `return` value of the function (method) to generate the comment for. If the return type cannot be detected through the static analysis of the function (method), evaluates to `void`.
- `${PARAM_DOC}` - - a documentation comment for parameters. Evaluates to a set of lines `@param type name`. If the function to generate comments for does not contain any parameters, the variable evaluates to empty content.
- `${THROWS_DOC}` - a documentation comment for exceptions. Evaluates to a set of lines `@throws type`. If the function to generate comments for does not throw any exceptions, the variable evaluates to empty content.
- `${DS}` - a dollar character (`$`). The variable evaluates to a plain dollar character (`$`) and is used when you need to escape this symbol so it is not treated as a prefix of a variable.
- `${CARET}` - indicated the position of the caret after generating and adding the comment.
  This `${CARET}` variable is applied only when a PHPDoc comment is generated and inserted during file creation. When a PHPDoc comment is created through Code | Generate | PHPDoc block, multiple selection of functions or methods is available so documentation comments can be created to several classes, functions, methods, or fields. As a result, PhpStorm cannot "choose" the block to apply the `${CARET}` variable in, therefore in this case the `${CARET}` variable is ignored.

Custom template variables

In addition to the predefined template variables, it is possible to specify custom variables. If necessary, you can define the values of custom variables right in a template using the `#set` VTL directive.

For example, if you want to use your full name instead of your login name defined through the pre-defined variable `${USER}`, write the following construct:

```
#set( $MyName = "John Smith" )
```

If the value of a variable is not defined in the template, PhpStorm will ask you to specify it when the template is applied.

You can prevent treating dollar characters (`$`) in template variables as prefixes. If you need a dollar character (`$` inserted as is, use the `${DS}` file template variable instead. When the template is applied, this variable evaluates to a plain dollar character (`$`).

Creating and Editing File Templates

PhpStorm supports several ways of creating file templates :

- Creating a file template from scratch
- Creating a file template from an existing one
- Creating a file template from a file
- Creating and referencing include templates

Creating a file template from scratch

1. Open Settings/Preferences dialog and under the Editor node, select File and Code Templates page .

2. Switch to the Files tab.

3. Click + on the toolbar and specify the template name, file extension, and the body of the template, which can contain:
   1. Plain text.
   2. `#parse` directives to work with includes .
   3. Predefined variables to be expanded into corresponding values in the format `${<variable_name>}` .
      The available predefined file template variables are:
      - `${PROJECT_NAME}` - the name of the current project.
      - `${FILE_NAME}` - the name of the PHP file that will be created.
      - `${NAME}` - the name of the new file which you specify in the New File dialog box during the file creation.
      - `${USER}` - the login name of the current user.
      - `${DATE}` - the current system date.
      - `${TIME}` - the current system time.
      - `${YEAR}` - the current year.
      - `${MONTH}` - the current month.
      - `${DAY}` - the current day of the month.
      - `${HOUR}` - the current hour.
      - `${MINUTE}` - the current minute.
      - `${PRODUCT_NAME}` - the name of the IDE in which the file will be created.
      - `${MONTH_NAME_SHORT}` - the first 3 letters of the month name. Example: Jan, Feb, etc.
      - `${MONTH_NAME_FULL}` - full name of a month. Example: January, February, etc.

      PhpStorm provides a set of additional variables for PHP include templates , that is, templates of reusable fragments that can be included in other PHP file template. The built-in PHP include templates are intended for generating file headers and PHPDoc documentation comments . The following variables are available in PHP include templates:
      - `${NAME}` - the name of the class, field, or function (method) for which the PHPDoc comment will be generated.
      - `${NAMESPACE}` - the fully qualified name (without a leading slash) of the class or field namespace.
      - `${CLASS_NAME}` - the name of the class where the field to generate the PHPDoc comment for is defined.
      - `${STATIC}` - gets the value `static` if the function (method) or field to generate the comment for is `static` . Otherwise evaluates to an `empty string` .
      - `${TYPE_HINT}` - a prompt for the `return` value of the function (method) to generate the comment for. If the return type cannot be detected through the static analysis of the function (method), evaluates to `void` .
      - `${PARAM_DOC}` - - a documentation comment for parameters. Evaluates to a set of lines `@param type name` . If the function to generate comments for does not contain any parameters, the variable evaluates to empty content.
      - `${THROWS_DOC}` - a documentation comment for exceptions. Evaluates to a set of lines `@throws type` . If the function to generate comments for does not throw any exceptions, the variable evaluates to empty content.
      - `${DS}` - a dollar character (`$`). The variable evaluates to a plain dollar character (`$`) and is used when you need to escape this symbol so it is not treated as a prefix of a variable.
      - `${CARET}` - indicated the position of the caret after generating and adding the comment.
        This `${CARET}` variable is applied only when a PHPDoc comment is generated and inserted during file creation. When a PHPDoc comment is created through Code | Generate | PHPDoc block, multiple selection of functions or methods is available so documentation comments can be created to several classes, functions, methods, or fields. As a result, PhpStorm cannot "choose" the block to apply the `${CARET}` variable in, therefore in this case the `${CARET}` variable is ignored.
   4. Custom variables. Their names can be defined right in the template through the `#set` directive or will be defined during the file creation.

4. To have the dollar character (`$`) in a variable rendered "as is", use the `${DS}` variable instead. This variable evaluates to a plain dollar character (`$`).

5. Apply the changes and close the dialog box.

Creating a file template from an existing one

1. Open the File Templates settings page and switch to the Files tab.

2. Click on the toolbar and change the template name, extension, and source code as required.

3. Apply the changes and close the dialog box.

Creating a file template from a file

1. Open the desired file in the editor.

2. On the main menu, choose Tools | Save File as Template.

3. In the File and Code Templates dialog box that opens specify the new template name and edit the source code, if necessary.

4. Apply the changes and close the dialog box.

Creating and referencing include templates

Include templates are used to define reusable pieces of code to be inserted in file templates through `#parse` directives. The built-in PHP include templates are intended for generating file headers and PHPDoc documentation comments . You can also define your own custom templates.

1. In the File and Code Templates settings page, switch to the Includes tab.

2. Click + on the toolbar and specify the template name, extension, and the source code. Do one of the following:
   - Use the predefined file template variables .
   - Create custom template variables and define their values right in the include template using the `#set` VTL directive.
     For example, if you want to your full name inserted in the file header instead of your login name defined through the `${USER}` , write the following construct:

     ```
     #set( $MyName = "John Smith" )
     ```

     If, when applying a template, the values of certain template variable are not known, PhpStorm will ask you to specify them.

   You can prevent treating dollar characters (`$`) in template variables as prefixes. If you need a dollar character (`$` inserted as is, use the `${DS}` file template variable instead. When the template is applied, this variable evaluates to a plain dollar character (`$`).

3. To use the include template, switch to the Templates tab, select the desired template and click Edit.

4. To include a template, insert the #parse directive in the source code.

Live Templates

In this part:

- Live Templates
  - Overview
  - Important notes
- Simple, Parameterized and Surround Live Templates
- Live Template Abbreviation
- Live Template Variables
- Groups of Live Templates
- Creating and Editing Live Templates
- Creating and Editing Template Variables
- Sharing Live Templates

Overview

Live templates let you insert frequently-used or custom code constructs into your source code file quickly, efficiently, and accurately. To expand a code snippet, type the corresponding template abbreviation and press `Tab`. Use `Tab` to jump from one variable in the template to another, see Live Template Variables .

Find the full list of predefined Live templates on the Live Templates page of the Settings / Preferences Dialog . For more details, see Creating Code Constructs by Live Templates and Creating Code Constructs Using Surround Templates .

Important notes

- To create or edit the Live templates, use the Live Templates page of the Settings/Preferences dialog.
- Having created a custom live template , you can export all the live templates together with the other settings, and import them. Refer to the section Exporting and Importing Settings for details.

Simple, Parameterized and Surround Live Templates

On this page:

- Simple live templates
- Parameterized live templates
- Surround live templates
- Examples

Simple live templates

Simple templates contain some fixed code that expands into plain text. When invoked and expanded in the editor, the code specified in the template is automatically inserted into your source code, replacing the abbreviation.

Parameterized live templates

Parameterized templates contain plain text and variables that enable user input.

> **Tip** If you need a dollar sign (`$`) in the template text, escape it by duplicating this character (`$$`).

After a template is expanded, variables appear in the editor as input fields whose values can be either filled in by the user or calculated by PhpStorm automatically.

When a parameterized live template is invoked and expanded in the Editor, PhpStorm can suggest some predefined values at the input positions of the variables. For example, if a parameterized template contains code for an iteration, then, on expanding the template, PhpStorm will suggest:

- A name for the index variable (`i` , `j` , etc.).
- A name for the assigned variable that holds the current container element.
- Type of the elements in the iterated container.

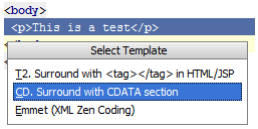
Surround live templates

Surround templates work only with blocks of selected text. Such templates place code before and after the selected block.

Examples

Insert parameterized live template (`Ctrl+J`):

Surround with live template (`Ctrl+Alt+J`):



Refer also to the page Wrapping a Tag. Example of Applying Surround Live Templates .

Live Template Abbreviation

Each live template is identified by a template abbreviation.

The template abbreviations work like shortcuts and are expanded into fragments of source code, depending on the surrounding context. So doing, PhpStorm can format the generated code fragments according to the code style settings.

An abbreviation may contain alphanumeric characters, dots, and hyphens, and must be unique within its group . However, you can use the same abbreviation for different templates provided that they are in different groups.

Whole code constructs can be created using the template abbreviations. To do that, type the template abbreviation, and press expansion key (for example, `Tab`). Refer to the section Creating Code Constructs by Live Templates for details.

Live Template Variables

On this page:

- What are template variables
- Declaring template variables
- Creating and editing template variables
- Predefined template variables
- Predefined functions to use in live template variables

What are template variables

Template variables in live templates enable user input. After a template is expanded, variables appear in the editor as input fields.

Declaring template variables

Variables within templates are declared in the following format:

```
$<variable_name>$
```

Creating and editing template variables

Variables are defined by expressions, and can accept some default values.

This expression may contain constructs of the following basic types:

- String constants in double quotes.
- The name of another variable defined in a live template.
- Predefined functions with possible arguments.

Template variables are editable in the Edit Template Variables Dialog , which contains a complete list of available functions. See the list of predefined functions below on this page.

Predefined template variables

PhpStorm supports two predefined live template variables: `$END$` and `$SELECTION$` .

You cannot edit the predefined live template variables `$END$` and `$SELECTION$` .

- `$END$` indicates the position of the cursor after the template is expanded. For example, the template `return $END$;` will be expanded into

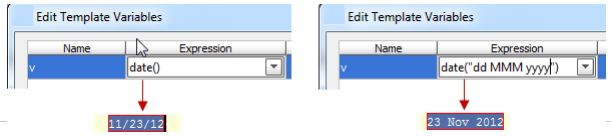
  ```
  return ;
  ```

  with the cursor positioned right before the semicolon.
- `$SELECTION$` is used in surround templates and stands for the code fragment to be wrapped. After the template is expanded, the selected text is wrapped as specified in the template.
  For example, if you select `EXAMPLE` in your code and invoke the `"$SELECTION$"` template via the assigned abbreviation or by pressing `Ctrl+Alt+T` and selecting the desired template from the list, PhpStorm will wrap the selection in double quotes as follows:

  ```
  "EXAMPLE"
  ```

Predefined functions to use in live template variables

Item	Description
`camelCase(String)`	Returns the string passed as a parameter, converted to camel case. For example, `my-text-file` / `my text file` / `my_text_file` will be converted to `myTextFile` .
`capitalize(String)`	Capitalizes the first letter of the name passed as a parameter.
`capitalizeAndUnderscore(sCamelCaseName)`	Capitalizes the all letters of a CamelCase name passed as a parameter, and inserts an underscore between the parts. For example, if the string passed as a parameter is `FooBar` , then the function returns `FOO_BAR` .
`classNameComplete()`	This expression substitutes for the class name completion at the variable position.
`clipboard()`	Returns the contents of the system clipboard.
`snakeCase(String)`	Returns CamelCase string out of snake_case string. For example, if the string passed as a parameter is `foo_bar` , then the function returns `fooBar` .
`complete()`	This expression substitutes for the code completion invocation at the variable position.
`completeSmart()`	This expression substitutes for the smart type completion invocation at the variable position.
`date(sDate)`	Returns the current system date in the specified format. By default, the current date is returned in the default system format. However, if you specify date format in double quotes, the date will be presented in this format:



`decapitalize(sName)`	Replaces the first letter of the name passed as a parameter with the corresponding lowercase letter.
`enum(sCompletionString1,sCompletionString2,...)`	List of comma-delimited strings suggested for completion at the template invocation.
`escapeString(sEscapeString)`	Escapes the specified string.
`expectedType()`	Returns the type which is expected as a result of the whole template. Makes sense if the template is expanded in the right part of an assignment, after return, etc.
`fileName(sFileName)`	Returns file name with extension.
`fileNameWithoutExtension()`	Returns file name without extension.
`firstWord(sFirstWord)`	Returns the first word of the string passed as a parameter.
`lineNumber()`	Returns the current line number.
`lowercaseAndDash(String)`	Returns lower case separated by dashes, of the string passed as a parameter. For example, the string `MyExampleName` is converted to `my-example-name`.
`snakeCase(sCamelCaseText)`	Returns snake_case string out of CamelCase string passed as a parameter.
`spaceSeparated(String)`	Returns string separated with spaces out of CamelCase string passed as a parameter. For example, if the string passed as a parameter is `fooBar`, then the function returns `foo bar`.
`time(sSystemTime)`	Returns the current system time.
`timestamp()`	This function without parameters returns the current time in milliseconds from the 1st January 1970. Refer to the article Unix time.
`underscoresToCamelCase(sCamelCaseText)`	Returns the string passed as a parameter with CamelHump letters substituting for underscores. For example, if the string passed as a parameter is `foo_bar`, then the function returns `fooBar`.
`underscoresToSpaces(sParameterWithSpaces)`	Returns the string passed as a parameter with spaces substituting for underscores.
`user()`	Returns the name of the current user.
`JsArrayVariable`	Returns JavaScript array name.
`jsClassName()`	Returns the name of the current JavaScript class.
`jsComponentType`	Returns the JavaScript component type.
`jsMethodName()`	Returns the name of the current JavaScript method.
`jsQualifiedClassName`	Returns the complete name of the current JavaScript class.
`jsSuggestIndexName`	Returns a suggested name for an index.
`jsSuggestVariableName`	Returns a suggested name for a variable.

Groups of Live Templates

On this page:

- Overview
- Managing groups of live templates

Overview

Live Templates are managed on the Live Templates page of the Settings / Preferences Dialog . For your convenience, templates are arranged in groups, most often in accordance with the context they are sensitive to.

PhpStorm comes with a set of predefined groups of templates. In addition to them, you can define your own custom groups and templates. It is strongly recommended that you avoid storing custom templates in predefined PhpStorm groups. For this purpose, use the user group, or a new group.

PhpStorm stores definitions of custom live template groups and templates added to predefined template groups in automatically generated configuration files `<group_name>.xml` .

- For a custom group, the file contains definitions of all the templates the group includes.
- For a customized predefined group, the file contains definitions of the added templates only.

Depending on the operating system you are using, the `<group_name>.xml` files are stored at the following locations:

- Windows: `<your_user_home_directory>.PhpStorm<version_number>\config\templates`
- Linux: `~PhpStorm<version>\config\templates`
- OS X: `~/Library/Preferences/PhpStorm<version>/templates`

Managing groups of live templates

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor. The Live Templates page that opens shows all the groups of live templates that are currently configured in PhpStorm.
2. Do one of the following:
   - To create a new custom group, click + on the toolbar and choose Template Group on the context menu. Then specify the name for the group in the Create New Group dialog box that opens.
   - To remove a group, select it in the list and click — on the toolbar.
   - To add a template to a group, select the group, click +, and choose Live Template on the context menu. In the Template editing area , specify the settings for the new template as described in Creating and Editing Live Templates .

Creating and Editing Live Templates

On this page:

- Introduction
- Modifying existing templates
- Creating a new live template from scratch
- Creating a new live template from a text fragment
- Searching through the list of live templates
- Restoring defaults

Introduction

PhpStorm comes with a set of the predefined Live Templates . You can use them as is, or modify them to suit your needs. If you want to create a new live template, you can do it from scratch, on the base of the copy of an existing template, or from a fragment of source code.

If a template has been changed, it is always possible to restore its default settings.

Modifying existing templates

To modify an existing template

1. In the Settings/Prefereces dialog, open the Live Templates page.
2. Expand the desired template group, and select the template you want to change.
3. In the Template Text area, change the template abbreviation as required.
4. In the Template Text field, edit the template body which may contain plain text and variables in the format `$<variable name>$` .
   When editing the live template variables, mind the following helpful hints:
   - If you need a dollar sign (`$`) in the template text, escape it by duplicating this character (`$$`).
   - To change the variables in a template, click the Edit Variables button and configure the variables as described in Creating and Editing Template Variables .

   The Edit Variables button is enabled only if the template body contains at least one user-defined variable, that is, a variable different from `$END$` or `$SELECTION$` .

   #### Side note about predefined template variables

   PhpStorm supports two predefined live template variables: `$END$` and `$SELECTION$` .

   You cannot edit the predefined live template variables `$END$` and `$SELECTION$` .
   - `$END$` indicates the position of the cursor after the template is expanded. For example, the template `return $END$;` will be expanded into

     ```
     return ;
     ```

     with the cursor positioned right before the semicolon.
   - `$SELECTION$` is used in surround templates and stands for the code fragment to be wrapped. After the template is expanded, the selected text is wrapped as specified in the template.
     For example, if you select `EXAMPLE` in your code and invoke the `"$SELECTION$"` template via the assigned abbreviation or by pressing `Ctrl+Alt+T` and selecting the desired template from the list, PhpStorm will wrap the selection in double quotes as follows:

     ```
     "EXAMPLE"
     ```
5. In the Options section, specify how the template will be expanded and reformatted.
6. In the Available in section, specify the languages and places of code where the editor should be sensitive to the template abbreviation.
7. Click OK when ready.

Creating a new live template from scratch

To create a new template from scratch

1. In the Settings dialog, open the Live Templates page, and expand the template group where you want to create a new template.
2. Click +. A new template item is added to the group and the focus moves to the Template Text area.
3. Specify the new template abbreviation, type the template body, define the variables and the template group, configure the options, as described in the template modification procedure.
4. Click OK when ready.

Creating a new live template from a text fragment

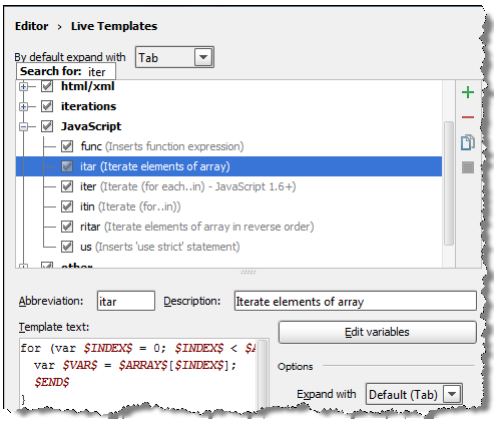
To create a live template from a text fragment

1. In the editor, select the text fragment to create a live template from.
2. On the main menu, choose Tools | Save as Live Template. The Live Templates page opens, with the Template Text area in focus.
3. In the Abbreviation field, type abbreviation to identify your new live template.
4. Specify the new template abbreviation, type the template body, define the variables and the template group, configure the options, as described in the template modification procedure.
5. Click OK when ready.

Searching through the list of live templates

To search through the list of live templates

– In the Live Templates page, start typing any string, which you expect to exist in the template abbreviation, body, or description.
PhpStorm shows all matching templates:



Restoring defaults

To restore default settings of a template

Note that a modified template is color-coded - it is shown blue.

1. In the Live Templates page, right-click a modified template to reveal its context menu.
2. Choose Restore defaults on the context menu of the modified template.

Creating and Editing Template Variables

On this page:

- Basics
- Configuring variables used in a template
- Predefined functions to use in live template variables

Basics

After a template is expanded, its variables are presented in the editor as input fields. The values of these fields can be either filled in by the user or calculated by PhpStorm.

To have it done automatically, for each variable you need to specify the following:

- Expression to be calculated in association with the variable.
- Default value to be entered in the input field if the calculation fails.

The order in which PhpStorm will process input fields after the template expansion, is determined by the order of variables in the list.

Configuring variables used in a template

1. Open the template settings , and in the Template Text area click the Edit Variables button.
   The Edit Variables button is enabled only if the template body contains at least one user-defined variable, that is, a variable different from `$END$` or `$SELECTION$` .

   The Edit Template Variables dialog box opens, where you can define how the variables will be processed when the template is used.

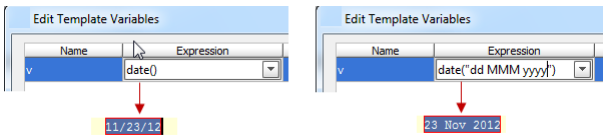
2. In the Name text box, specify the variable name to be used in the template body.

3. In the Expression drop-down list, specify the expression to be calculated by PhpStorm when the template is expanded. Do one of the following:
   - Type a string constant in double quotes.
   - Type a predefined function with possible arguments or select one from the drop-down list.
     An argument of a function can be either a line constant or another predefined function. See the list of predefined functions below on this page.

4. To enable PhpStorm to proceed with the next input field, if an input field associated with the current variable is already defined, select the Skip if defined check box.

5. To arrange variables in the order you want PhpStorm to switch between associated input fields, use the Move Up and Move Down buttons.

Predefined functions to use in live template variables

Item	Description
`camelCase(String)`	Returns the string passed as a parameter, converted to camel case. For example, `my-text-file` / `my text file` / `my_text_file` will be converted to `myTextFile` .
`capitalize(String)`	Capitalizes the first letter of the name passed as a parameter.
`capitalizeAndUnderscore(sCamelCaseName)`	Capitalizes the all letters of a CamelCase name passed as a parameter, and inserts an underscore between the parts. For example, if the string passed as a parameter is `FooBar` , then the function returns `FOO_BAR` .
`classNameComplete()`	This expression substitutes for the class name completion at the variable position.
`clipboard()`	Returns the contents of the system clipboard.
`snakeCase(String)`	Returns CamelCase string out of snake_case string. For example, if the string passed as a parameter is `foo_bar` , then the function returns `fooBar` .
`complete()`	This expression substitutes for the code completion invocation at the variable position.
`completeSmart()`	This expression substitutes for the smart type completion invocation at the variable position.
`date(sDate)`	Returns the current system date in the specified format. By default, the current date is returned in the default system format. However, if you specify date format in double quotes, the date will be presented in this format: 
`decapitalize(sName)`	Replaces the first letter of the name passed as a parameter with the corresponding lowercase letter.
`enum(sCompletionString1,sCompletionString2,...)`	List of comma-delimited strings suggested for completion at the template invocation.
`escapeString(sEscapeString)`	Escapes the specified string.
`expectedType()`	Returns the type which is expected as a result of the whole template. Makes sense if the template is expanded in the right part of an assignment, after return, etc.
`fileName(sFileName)`	Returns file name with extension.
`fileNameWithoutExtension()`	Returns file name without extension.
`firstWord(sFirstWord)`	Returns the first word of the string passed as a parameter.
`lineNumber()`	Returns the current line number.
`lowercaseAndDash(String)`	Returns lower case separated by dashes, of the string passed as a parameter. For example, the string `MyExampleName` is converted to `my-example-name` .
`snakeCase(sCamelCaseText)`	Returns snake_case string out of CamelCase string passed as a parameter.

`spaceSeparated(String)`	Returns string separated with spaces out of CamelCase string passed as a parameter. For example, if the string passed as a parameter is `fooBar`, then the function returns `foo bar`.
`time(sSystemTime)`	Returns the current system time.
`timestamp()`	This function without parameters returns the current time in milliseconds from the 1st January 1970. Refer to the article Unix time.
`underscoresToCamelCase(sCamelCaseText)`	Returns the string passed as a parameter with CamelHump letters substituting for underscores. For example, if the string passed as a parameter is `foo_bar`, then the function returns `fooBar`.
`underscoresToSpaces(sParameterWithSpaces)`	Returns the string passed as a parameter with spaces substituting for underscores.
`user()`	Returns the name of the current user.
`JsArrayVariable`	Returns JavaScript array name.
`jsClassName()`	Returns the name of the current JavaScript class.
`jsComponentType`	Returns the JavaScript component type.
`jsMethodName()`	Returns the name of the current JavaScript method.
`jsQualifiedClassName`	Returns the complete name of the current JavaScript class.
`jsSuggestIndexName`	Returns a suggested name for an index.
`jsSuggestVariableName`	Returns a suggested name for a variable.

Sharing Live Templates

On this page:

- Configuration files with definitions of custom live templates
- Sharing live templates manually through configuration files
- Sharing live templates through export/import
  - Example of sharing templates among different IDE
- Sharing live templates among template groups

Configuration files with definitions of custom live templates

PhpStorm stores definitions of custom live template groups and templates added to predefined template groups in automatically generated configuration files `<group_name>.xml`.

- For a custom group, the file contains definitions of all the templates the group includes.
- For a customized predefined group, the file contains definitions of the added templates only.

Depending on the operating system you are using, the `<group_name>.xml` files are stored at the following locations:

- Windows: `<your_user_home_directory>.PhpStorm<version_number>\config\templates`
- Linux: `~PhpStorm<version>\config\templates`
- OS X: `~/Library/Preferences/PhpStorm<version>/templates`

Sharing live templates manually through configuration files

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor.
2. Create the required custom template groups and update the relevant predefined groups as necessary and click OK. Based on these changes, PhpStorm generates the `<group_name>.xml` files, see Location of Custom Live Templates Definitions above.
3. Locate the generated `<group_name>.xml` files and do one of the following:
   - To share the templates among your teammates, send the relevant files to them with the instruction to save the files in the `templates` folder.
   - To use the templates in another PhpStorm installation on your computer, copy the relevant files to the `templates` folder under the relevant `PhpStorm<version>` folder.

Sharing live templates through export/import

PhpStorm allows you to easily share live templates among your team members, numerous PhpStorm installations, and even different IDE by using the standard Export/Import functionality. You can share custom template groups and updates to predefined groups.

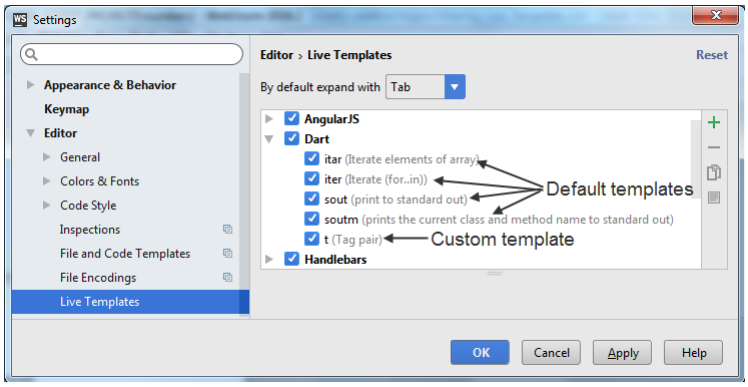
1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor.
2. Create the required custom template groups and update the relevant predefined groups as necessary and click OK. Based on these changes, PhpStorm generates the `<group_name>.xml` files, see Location of Custom Live Templates Definitions above.
3. On the main menu, choose File | export Settings.
4. In the Export Settings dialog box that opens, select the Live Template check box and specify the name of the `.jar` file where the exported settings will be stored. When you click OK, PhpStorm generates a file with the specified named based on the `.xml` configuration files stored in the `templates` folder.
5. Do one of the following:
   - To share the templates among your teammates, pass the generated `.jar` file to them with the following instructions:
     1. Save the received `.jar` file on your computer.
     2. Choose File | Import Settings on the main menu and and specify the location of the received `.jar` file.
     3. In the Select Components to Import dialog box that opens, select the Live Templates check box and click OK.

     PhpStorm restarts whereupon the imported templates are displayed on the Live Templates page.
   - To use the templates in another PhpStorm installation or in another IDE on your computer, open the required installation, choose File | Import Settings on the main menu, and specify the location of the generated `.jar` file.

Example of sharing templates among different IDE

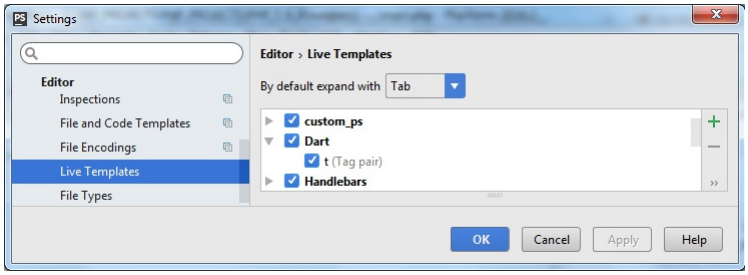
Be careful when sharing templates among different IDE. If you import custom templates (updates) from a group which is predefined in the source IDE but is not predefined in the target IDE, such group will be created but will contain only the custom templates. The example below shows what happens if we add a template to a predefined group in WebStorm and then reuse it in PhpStorm.

In WebStorm, the Dart template group is predefined. If we add the `t (tag pair)` template to it, this update will be saved in the `Dart.xml` file:



In PhpStorm, there is no predefined template group Dart. So when we export the live templates from WebStorm and them import them into PhpStorm, a Dart group is

created but it contains only one template `t (tag pair)`, which we added to the group in WebStorm before export:



Sharing live templates among template groups

You can copy and move templates from one group to another.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor.
2. Do one of the following:
   - To copy a template to another group:
     1. Select the template in interest. Use `Ctrl` and `Shift` keys for multiple selection.
     2. Choose Copy on the context menu of the selection.
     3. Select the group to copy the template to and choose Padte on the context menu of the selection.
   - To move a template to another group, select the required template, choose Move on the context menu of the selection and choose the group to move the template to.

Populating Projects

In this part:

- Populating Projects
  - Basics
- Creating Directories
- Creating Files from Templates
- Creating Empty Files
- Creating PHP Classes, Interfaces, and Traits

Basics

Populate your project by creating new elements of various types (directories and files). PhpStorm suggests the following alternative ways of accessing the corresponding functionality:

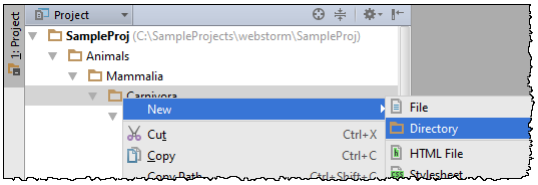
- File | New on the main menu.
- The New context menu command.
- The keyboard shortcut `Alt+Insert`.

As a result, the New menu is shown in which you select the type of an element to be created.

Creating Directories

PhpStorm enables you to create directories recursively, thus producing a whole directory structure.

1. Open the Project tool window (e.g. View | Tool Windows | Project).
2. Select the destination directory.
3. Do one of the following:
   - Select File | New | Directory.
   - Select New | Directory from the context menu.
   - Press `Alt+Insert` and select Directory.



4. In the dialog that opens, specify the directory name. If you want to create a number of nested directories, specify the directory names separated with slashes.

Creating Files from Templates

On this page:

– Introduction

– Creating a new file from a template

Introduction

PhpStorm provides file templates for most of the languages that it supports . This lets you create the files with the initial content appropriate for the file purpose. For example, there are file templates for HTML/HTML5/XHTML, XML, and JavaScript files, PHP classes and files, XSLT style sheets, and other supported file types.

Generally, the file name extension for a template-based file is set automatically so you don't need to specify it. For example, if you create a JavaScript file, it gets the `.js` extension. New HTML and XML files get the `.html` and `.xml` extensions respectively. New XSLT style sheets get the `.xsl` extension.

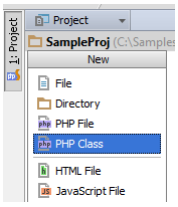
Sometimes, you are given an opportunity to select the desired extension from the list. This, for example, is the case when creating PHP classes.

Creating a new file from a template

To create a new file from a template

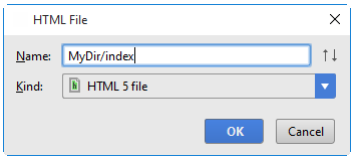
1. Do one of the following:
   – In the Project tool window , select the directory in which you want to create a new file, and then choose File | New on the main menu.
   – Right-click the corresponding directory and select New from the context menu.
   – Press `Alt+Insert`

2. Select the desired file type. Generally, all the options except File and Directory correspond to using a file template.



   An existing file template may be missing from the list if this is a custom template whose file name extension (template extension) does not match registered patterns of any of the recognized file types. In such a case, you may want to register the corresponding pattern for an existing recognized file type or add a new file type and register the corresponding pattern for this new type. For more information, see Creating and Registering File Types .

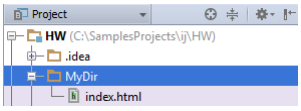
3. In the dialog that opens, type the name of the file in the corresponding field. Note that you should not type the file name extension.
   You can specify the whole directory structure prepending the new file name. If the nested directories do not yet exist, they will be created:



   If required, specify the kind of the new template-based file. For example, if you select to create a class, you'll be able to create a class, interface, etc. or, in other words, use one of the corresponding related file templates.

   Specify other information as required. For example, you may be asked to define the values of custom variables if the corresponding file template contains such variables and their values are not currently set.

4. Click OK. The new file that corresponds to the selected file template will be created under the target location.
   If the names of non-existent sub-directories were specified before the new file name, the whole structure will be created under the target directory:



Sometimes, you may want to change the auto-generated file name extension. To do that, use the Rename refactoring (Refactor | Rename).

Creating Empty Files

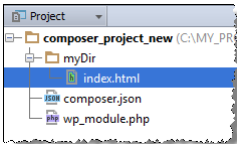
Generally, all the files that you create when developing applications are template-based . However, sometimes you may want to create empty files.

To create an empty file

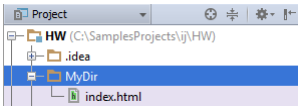
1. Do one of the following:
   - In the Project tool window , select the directory in which you want to create a new file, and then choose File | New on the main menu.
   - Right-click the corresponding directory and select New from the context menu.
   - Press `Alt+Insert`

2. Select File.

3. In the New File dialog, in the field under Enter a new file name, type the file name and extension.
   You can specify the whole directory structure prepending the new file name. If the nested directories do not yet exist, they will be created:



4. Click OK.
   If the names of non-existent sub-directories were specified before the new file name, the whole structure will be created under the target directory:



5. If the extension you have specified is not associated with any of the file types recognized by PhpStorm, the Register New File Type Association dialog is displayed. In this dialog, you can associate the extension with one of the recognized file types. To do that, select the file type under Open matching files in PhpStorm and click OK. As a result, the extension is associated with the specified file type.

> **Tip** If there are no appropriate file types for the new extension, you may want to create a new file type and associate the extension with that type. For more information, see Creating and Registering File Types .

Creating PHP Classes, Interfaces, and Traits

On this page:

- Basics
- Creating PHP classes

Basics

PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when creating classes, interfaces, and traits.

When you specify the folder where the file that implements the new class will be stored, PhpStorm automatically updates the namespace for the class.

Creating PHP classes

1. In the Project tool window, select the folder that corresponds to the namespace where you want a new class/interface/trait created and then choose New | PHP Class on the context menu of the selection. The Create New PHP Class dialog box opens.
2. In the Name text box, type the name of the class/interface/trait to be created. PhpStorm automatically fills in the specified name in the File Name field.
3. Specify the namespace to create the class/interface/trait in. By default, the Namespace field shows the namespace that corresponds to the folder from which the class/interface/trait creation was invoked.
   You can choose Global namespace from the drop-down list or change the namespace manually keeping mind that this may cause problems with loading as such namespace hierarchy will not meet the PSR0 standards.

   You can also change the folder to store the file that will implement the new class/interface/trait, the change will be also reflected in the Namespace field.
4. In the Kind drop-down list, specify what to create, the available options are:
   - Class
   - Interface
   - Trait
5. Choose the file extension from the drop-down list.

When you click OK, a new class is created according to the PHP Class template with the specified namespace declaration automatically added.

Generating Code

This section describes how to quickly and accurately populate your source code with the complicated code constructs.

In this part:

- Creating Code Constructs by Live Templates
- Creating Code Constructs Using Surround Templates
- Generating Getters and Setters
- Unwrapping and Removing Statements
- Wrapping a Tag. Example of Applying Surround Live Templates
- Rearranging Code Using Arrangement Rules
- Surrounding Blocks of Code with Language Constructs
- Using Postfix Templates

Creating Code Constructs by Live Templates

On this page:

- Introduction
- Inserting a live template

Introduction

This page describes how to generate source code using live templates .

Using Live Templates enables you to create such code constructs as JavaScript iterations, Emmet, HTML and XML tags.

To explore the list of available live templates, in the Settings/Preferences dialog, open the Live Templates page.

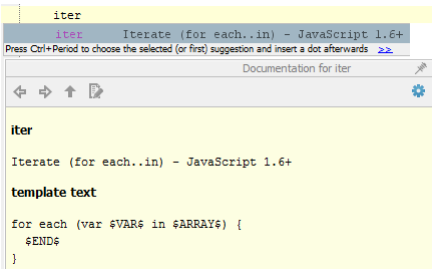
Inserting a live template

To insert a live template in the source code

1. Place the caret at the desired position, where the new construct should be added.

2. Do one of the following:
   - On the main menu, choose Code | Insert Live Template.
   - Press `Ctrl+J`.
   - Type some initial letters of the template abbreviation to get the matching abbreviations in the suggestion list. Note that the suggestion list may contain same abbreviations for different templates.

3. From the suggestion list , select the desired template. While the suggestion list is displayed, it is possible to view Quick Documentation for the items at caret, by pressing `Ctrl+Q`.



4. Press the template invocation key (this may be `Space`, `Tab` or `Enter`, depending on the template definition). The new code construct is inserted in the specified position.

5. If the selected template is parametrized and requires user input , the editor enters the template editing mode and displays the first input field highlighted with the red frame. Type your value in this frame and press `Enter` or `Tab` to complete input and pass to the next input field. After completing the last input field, the caret moves to the end of the construct, and the editor returns to the regular mode of operation.

It is also possible to type a template abbreviation, and then press `Ctrl+J`.

Creating Code Constructs Using Surround Templates

This section describes how to wrap fragments of code according to surround templates .

To surround a block of code with a live template

1. In the editor, select the piece of code to be surrounded.

2. Do one of the following:
   - On the main menu, choose Code | Surround With Live Template....
   - Press `Ctrl+Alt+J`.

3. Select the desired template from the suggestion list.

Generating Getters and Setters

On this page:

- Introduction
- Generating accessor and mutator methods

Introduction

In the PHP context, you can generate accessor and mutator methods (getters and setters) for the fields of in your classes. PhpStorm generates getters and setters with only one argument.

In the PHP context, getters and setters are generated using the PHP getter/setter file template. By default, as specified in these templates, setters are generated with the `set` prefix and getters with the `set` or `get` prefix according to the inferred field type `boolean` or `con-boolean`. The prefix is the value of the `${GET_OR_IS}` variable in the default getter template. The default template is configured in the Code tab on the File and Code Templates page of the Settings / Preferences Dialog .

Generating accessor and mutator methods

1. On the main menu, choose Code | Generate.

   Alternatively, right-click the editor and choose Generate on the context menu, or use `Alt+Insert` keyboard shortcut.
2. In the pop-up list that is displayed in the editor, select one of the following options:
   - Getter: Accessor methods for getting the current values of the fields that will be selected in the Choose Fields to Generate Getters and Setters dialog box.
   - Setter: Mutator methods for setting specified values to the fields.
   - Getter and Setter: Both methods.
3. In the Choose Fields to Generate Getters and Setters dialog box, select the fields to generate getters or setters for.
4. Click OK when ready.

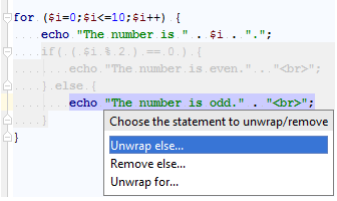
Unwrapping and Removing Statements

PhpStorm enables you to quickly unwrap or extract expressions from the enclosing statements. This action is available for:

- JavaScript: if ...else , for , while , and do...while control structures.
- XML and HTML tags.
- PHP: if , else , while , do...while , for , and try...catch control structures

To unwrap or remove a statement

1. Place the caret on the expression you want to extract or unwrap.
2. Choose Code | Unwrap/Remove on the main menu or press `Ctrl+Shift+Delete`. PhpStorm shows a pop-up window with all the actions that are available in the current context. Statements to be extracted are displayed on the blue background, statements to be removed are displayed on the grey background.



3. Click the desired action or select it using the up and down arrow keys and press `Enter`.

Wrapping a Tag. Example of Applying Surround Live Templates

As an example of applying a surround template, let's wrap a piece of XML code with tags.

To surround a code fragment

1. Open the desired file for editing.

2. Select a code fragment.

```
<company>
    <name>Peter</name>
    <name>John</name>
    <name>Sarah</name>
</company>
```

3. Press invocation shortcut `Ctrl+Alt+J`. PhpStorm suggests the following surround templates:

```
    <company>
        <name>Peter</name>
        <name>John</name>
        <name>Sarah</name>
    </company>
Select Template
T Surround with <tag></tag>
```

4. Select the tag template from the suggestion list. The code fragment is surrounded with empty tags:

```
<company>
    <>
        <name>Peter</name>
        <name>John</name>
        <name>Sarah</name>
    </>
</company>
```

5. The caret rests within the opening one. On typing the tag name in the opening tag, the name is automatically reproduced in the closing tag:

```
<company>
    <staff>
        <name>Peter</name>
        <name>John</name>
        <name>Sarah</name>
    </staff>
</company>
```

Rearranging Code Using Arrangement Rules

On this page:

– Defining arrangement rules

– Rearranging code

Defining arrangement rules

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Code Style under Editor.
2. Under the Code Style node, select the appropriate language.
3. On the Arrangement tab of the language settings, specify the rules for your code rearrangement and click OK.

Rearranging code

1. In the editor select a part of the code that you want to rearrange.
2. On the main menu, choose Code | Rearrange Code.

   Alternatively, in the project tool window, right-click the file where you want to perform the code rearranging, select Code | Reformat Code.
3. In the dialog that opens, select Rearrange entries check box and click OK.

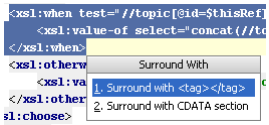
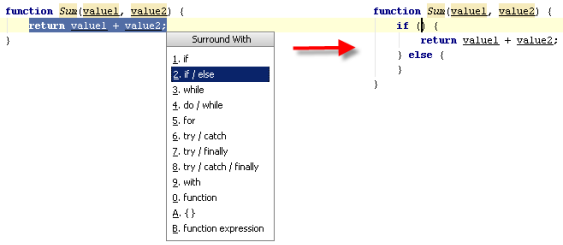
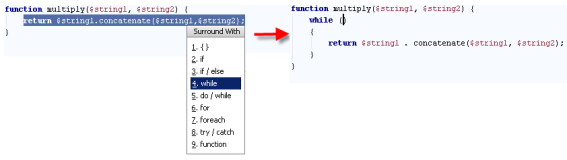
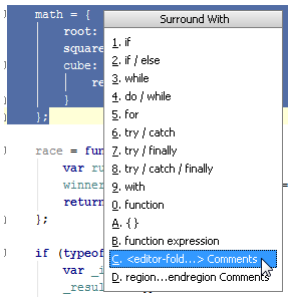
Surrounding Blocks of Code with Language Constructs

On this page:

- Applicable contexts
- Surrounding blocks of code

Applicable contexts

The Surround with feature allows you easily put expressions or statements within blocks or language constructs. This feature in PhpStorm applies to:

Context	Surround with	Example
XML/HTML tags	– Tag – CDATA section – <% ... %> – Emmet	
JavaScript statements	– `(expr)` – `!(expr)` – `if` – `if / else` – `while` – `do / while` – `for` – `try / catch` – `try / finally` – `try / catch / finally` – `with` – `function` – `{ }` – `function expression`	
PHP statements	– `if` – `if / else` – `while` – `do / while` – `for` – `foreach` – `try / catch` – `function`	
Code Folding	Any fragment of code, where Surround With is applicable	

Surrounding blocks of code

To surround a block of code

1. Select the desired code fragment.
2. Do one of the following:
   - On the main menu, choose Code | Surround With
   - Press `Ctrl+Alt+T`.

   A pop-up window displays the list of enclosing statements according to the context.
3. Select the desired surround statement from the list. To do that, use the mouse cursor, up and down arrow keys, or a shortcut key displayed next to each element of the list.

Using Postfix Templates

On this page:

– Overview

– Activating and de-activating postfix templates

– Using postfix templates

Overview

Postfix code completion lets you transform an already typed expression to another one based on the postfix you type after a dot, the type of the expression, and its context. This transformation is performed by expanding the postfix-specific predefined template. For example, the `.if` postfix applied to a boolean expression wraps it with an `if` statement. See more at: Postfix Code Completion .

Activating and de-activating postfix templates

When the postfix template functionality is enabled, PhpStorm transforms expressions with postfixes into other expressions by expanding the corresponding postfix-specific templates.

To activate the postfix templates functionality:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Postfix under General.
   The Postfix Completion page that opens shows a list of postfixes for which PhpStorm can transform expressions according to postfix-specific templates. Transformation templates are shown next to corresponding postfixes. When you select a postfix, the Description pane illustrates the corresponding transformation showing the code snippet before and after the template is expanded.
2. Select the Enable postfix templates check box and choose the key to launch template expansion from the Expand template with drop-down list.
3. Select the check boxes next to the postfixes for which you want PhpStorm to expand the corresponding templates.

To suppress expanding a template, clear the check box nex to the corresponding postfix. To block the postfix templates functionality, clear the Enable postfix templates check box.

Using postfix templates

To apply a postfix template:

1. Type your code followed by a dot, until a suggestion list appears, or press `Ctrl+J`.
2. Choose the desired postfix template from the suggestion list. If necessary, choose an expression to be surrounded or replaced.
3. Press the expansion key, defined in the Postfix Completion page of the editor settings.

Auto-Completing Code

This part describes the various techniques that enable you to speed up the editing process. These techniques include:

- Basic code completion on `Ctrl+Space`.
- Type completion on `Ctrl+Shift+Space`.
- Completing punctuation on `Enter`.
- Completing statements with smart `Enter`.
- Completing paths in the Select Path dialog .
- Expanding words with `Alt+Slash`.
- Various tips and tricks on using the suggestion lists that appear on invoking code completion.

Tip All types of code completion offered by PhpStorm support "CamelHumps" and "Middle Matching". It means that you can either complete any item by entering only its uppercase characters, or start typing any part of an item.

All completion results are ranged and ordered by relevance considering the caret position and surrounding code.

Basic Code Completion. Completing Names and Keywords

On this page:

- Basics
- Completing names
- Tips and tricks

Basics

Basic code completion helps you complete the names of classes, methods, and keywords within the visibility scope. When you invoke code completion, PhpStorm analyses the context and suggests the choices that are reachable from the current position of the caret.

If basic code completion is applied to a part of a parameter, or variable declaration, PhpStorm suggests a list of possible names with regard to the type of the item.

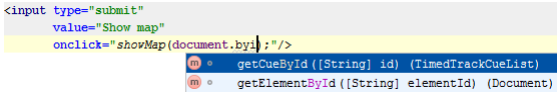
> **Note** PhpStorm recognizes variables declared in PHP-specific ways, for example, as parameters in function calls or defined by reference argument.

Code completion covers supported file types and custom file types. However, PhpStorm does not recognize the structure of custom file types and suggests completion options regardless of whether a specific type is appropriate in the current context.

Completing names

To automatically complete names in the current visibility scope, follow these general steps:

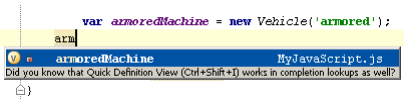
1. Type a name or a part of a name.
   While typing, use the following techniques:
   - Narrow down the suggestion list by typing any part of a word, even the characters somewhere in the middle, or invoking code completion after a dot separator. PhpStorm will show suggestion list that includes occurrences with all the possible positions of the entered characters.
     This makes the use of the asterisk wildcard unnecessary:



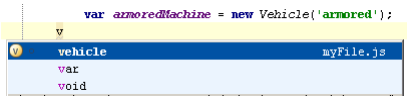
   - In case of CamelCase or snake_case names, type the initial letters only. So doing, PhpStorm automatically recognizes and matches the initial letters.

2. Do one of the following:
   - On the main menu, choose Code | Completion | Basic.
   - Press `Ctrl+Space`.

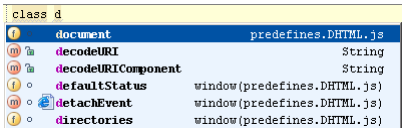
   The image below shows a suggestion list containing a variable name:



   The image below shows a suggestion list containing keywords:



3. If necessary, press `Ctrl+Space` for the second time.



> **Tip** This action produces the same effect as pressing `Ctrl+Alt+Space`.

4. Select the desired class from the suggestion list, and do one of the following:
   - Press `Enter` or double-click the desired choice .
   - Press `Tab` .

Tips and tricks

- If a suggestion list shows n variants of a class, type a dot, and the next suggestion list after dot will show all the possible qualified names of the members. After selecting the desired one, import statement will be added automatically.
- Suggestion list includes literals, already defined in a file.

Smart Type Code Completion. Completing Code Based on Type Information

On this page:

- Introduction
- Using smart type code completion

Introduction

Smart Type code completion filters the suggestion list and includes only those types that are applicable to the current context.

Using smart type code completion

To use smart type code completion

1. Do one of the following:
   - On the main menu, choose Code | Complete Code | SmartType.
   - Press `Ctrl+Shift+Space`.

   SmartType code completion automatically highlights selection in the suggestion list that seems most suitable for the current context. Such results display on a darker green background.

   As you type, the suggestion list reduces to show matching entries only.

2. If necessary, press `Ctrl+Shift+Space` once more.

Completing Statements

On this page:

- Completing statements
- Examples
  - Completing code constructs

Completing statements

The Smart Enter completion enables you to create syntactically correct code constructs.

To automatically complete a statement, follow these steps:

- Start typing a statement, and press `Ctrl+Shift+Enter`.
  The punctuation required in the current context is added and the caret moves to the next editing position.

Examples

Completing code constructs

Start typing a code construct:

```
function getVehicleInformation() {
    if
}
```

(expected

After pressing `Ctrl+Shift+Enter`, the construct is completed, and the caret is placed at the next editing position:

```
function getVehicleInformation() {
    if ()
}
```

expression expected
statement expected

After pressing Ctrl+Shift+Enter

Completing Path

On this page:

- Introduction
- Completing path

Introduction

This type of completion aims at speeding up selection of files, folders, etc. The dialog used for that purpose is called rather often, for example when configuring a PHP interpreter .

Completing path

To automatically complete a path

1. Open the Select Path dialog by doing one of the following:
   - Click the Browse button .
   - Choose Open File on the File menu.
2. Make sure that the Path field is visible. If the field is hidden, click the Show path link.
3. Start typing a path. PhpStorm suggests a drop-down list where you can quickly find the needed directory. As you type, the suggestion list shrinks to show the matching paths only. You can also press `Ctrl+Space` to show the suggestion list.
4. Select completion from the list.

Completing Punctuation

On this page:

- Introduction
- Configuring paired elements behavior

Introduction

In the Editor , you can configure the editor's behavior related to the paired braces. When the option Insert pair } is enabled, the closing brace is automatically added with indent on pressing `Enter` . Indentation level is defined in the Code Style settings .

Configuring paired elements behavior

To configure insertion of paired elements

1. Open the Settings dialog box , and under the Editor node, click Smart Keys .
2. Configure automatic insertion of paired elements by selecting the following check boxes:
   - Insert pair bracket
   - Insert pair quote
   - Auto-close tag on typing '</'
3. To have the closing braces inserted upon pressing `Enter` , select the Insert pair } check box.
4. Apply changes and click OK.

Completing Tag Names

On this page:

- Introduction
- Completing tag names
- Inserting taglib declaration
- Importing a taglib declaration

Introduction

PhpStorm automatically completes the names of tags, attributes and their values in files of the following types:

- HTML/XHTML
- XML/XSL

Automatic tag name completion is based on the DTD or Schema the file is associated with.

If there is no schema association, PhpStorm will use the file content (tag and attribute names and their values) to complete your input.

In XML/XSL files, completion for taglibs and namespaces is available.

You can have PhpStorm do one of the following:

- Insert a declaration of the taglib the tag in question belongs to.
- Import the desired taglib and insert all required import and reference statements.

Completing tag names

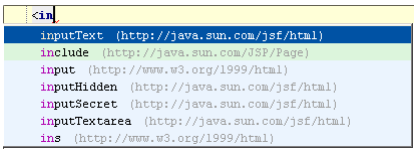
To invoke the tag name completion

1. Press `<` and start typing the tag name. PhpStorm displays the list of tag names appropriate in the current context. Use the `ArrowUp` and `ArrowDown` buttons to scroll through the list.
2. Press `Enter` to accept selection from the list, or continue typing the tag name manually.
   PhpStorm automatically inserts mandatory attributes according to the schema.

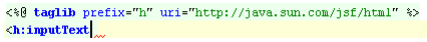
Inserting taglib declaration

To insert a taglib declaration

1. Start typing the tag and press `Ctrl+Alt+Space`.
2. From the list of matching tags, select the desired one. The `uri` of the taglib it belongs to is displayed in brackets.



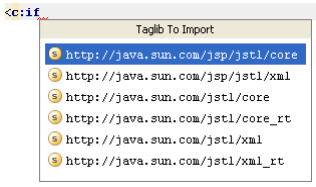
3. Select the desired taglib and press `Enter`. PhpStorm adds the declaration of the selected taglib.



Importing a taglib declaration

To import a taglib

1. Start typing the taglib prefix and press `Alt+Insert`.
2. Select the desired taglib from the list of suggestions and press `Enter`.



PhpStorm imports the selected taglib and adds the import statement automatically.

Hippie Completion. Expanding Words

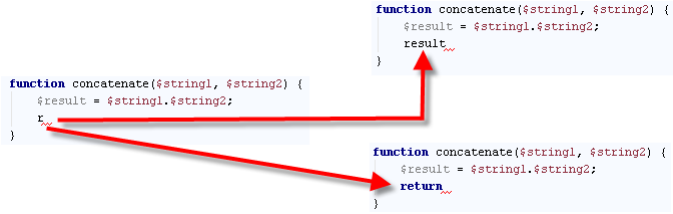
This helpful feature analyses your text in the visible scope and suggests to complete a word with a keyword, class name, method or variable. So doing, the prototype is highlighted in the source code.

To expand a string at caret to an existing word

1. Type the initial string.
2. Do one of the following:
   - On the main menu, choose Code | Completion | Cyclic Expand Word.
   - Press `Alt+Slash`.

   The editor shows the first suggested value.
3. Press `Enter` to accept the suggested word, or continue pressing `Alt+Slash` until the desired word is found.

Example:



Using Suggestion List

On this page:

- Introduction
- Navigating through the suggestion lists
- Narrowing and widening the suggestion list
- Completing selection
- Viewing reference information in the suggestion lists
- Sorting entries in a suggestion list
- Viewing hierarchy in a completion pop-up
- Closing a suggestion list

Introduction

Any code completion or live template operation ends up with a suggestion list, from which you have to select the desired option.

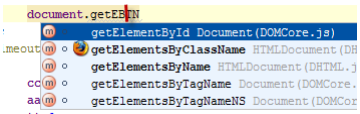
PhpStorm provides handy ways to make your selection, complete the source code, sort entries, and view reference information without leaving the suggestion list.

Navigating through the suggestion lists

To navigate through the suggestion lists, use the mouse cursor or the `Up` and `Down` arrow keys.

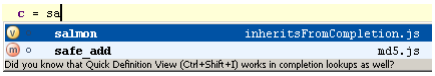
Narrowing and widening the suggestion list

- Use the horizontal arrow keys in the completed expression to make the suggestion list wider (left arrow) or closer (right arrow):

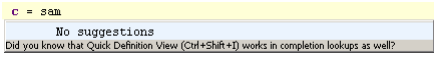


Completing selection

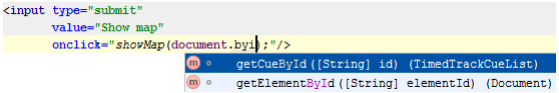
- Use `Enter` to insert the selected string at the insertion point.
- Use `Tab` to replace the string next to the caret with the selected one.
- Use `Ctrl+Shift+Enter` to make the current code construct syntactically correct (balance parentheses, add missing braces and semicolons, etc.)
- It is possible to enter the desired element name manually when the input position is highlighted and the list of suggestions is opened. As you type, the suggestion list shrinks so that only matching element names remain:



If you enter a name not used in scope, PhpStorm informs you with a pop-up window, as shown in the figure below:



- It is possible to type any part of a word, even the characters located somewhere in the middle:



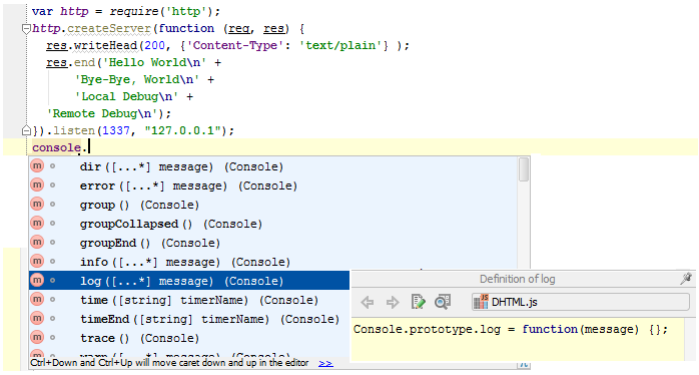
That's why asterisk character doesn't get any special processing as a wildcard.

Please note the following:

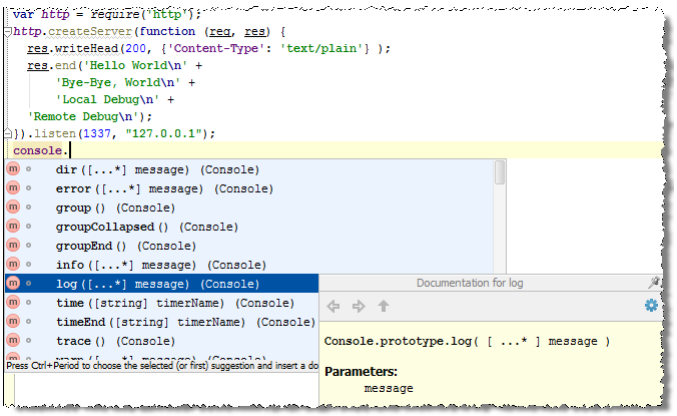
- If, invoking an action, you don't get what you want immediately, invoke this action again by pressing the same shortcut .
- Most suitable variants in the suggestion list are displayed on top of the list.
- Entries in bold font on top of the list denote symbols defined in the current class. All the other symbols are shown below.

Viewing reference information in the suggestion lists

- Use the Quick Definition View . If you select an entry in a list and press `Ctrl+Shift+I`, PhpStorm displays the following quick information pop-up window:



- Quick Information View works in a suggestion list. If you select an entry in a list, and press `Ctrl+Q`, PhpStorm displays the following quick information pop-up window:



Sorting entries in a suggestion list

To sort entries in a suggestion list:

– Click A or π in the lower-right corner of the list to toggle between sorting in alphabetical order, or by relevance.

Please note the following:

– The sorting icon appears in the list if it is long enough. For few entries only sorting icon is not displayed.

– Use mouse only to click this icon.

– PhpStorm memorizes the type of sorting.

– Default behavior is defined in the Code Completion page of the Editor settings.

Viewing hierarchy in a completion pop-up

To view hierarchy in a completion pop-up, follow these steps:

1. Start typing the source code, and press `Ctrl+Shift+Space` or `Ctrl+Space`. The suggestion list appears.
2. While in the suggestion list, press `Ctrl+H`. The Hierarchy tool window shows the type hierarchy of the class, selected in the suggestion list.

Closing a suggestion list

Close suggestion lists with `Escape`, then enter your own code at the code completion position.

Creating and Optimizing Imports

In this part:

- Creating Imports
- Optimizing Imports

Creating Imports

On this page:

- Introduction
- Completing a short class name and importing a PHP namespace on-the-fly
- Importing a PHP namespace using a quick fix
- Importing an XML namespace
- Importing TypeScript symbols

Introduction

The import statement adds to the imports section, but the cursor does not move from the current position, and your current editing session does not interrupt. This feature is known as the Import Assistant.

> **Tip** To configure the way auto-import works, open the PhpStorm settings , and then go to the Editor: Auto Import page.

The same possibility applies to the XML files. When you type a tag with an unbound namespace , import assistant suggests to create a namespace and offers a list of appropriate choices.

When you reference a PHP class that is defined outside the current file, PhpStorm locates the class definition, whereupon you can do one of the following:

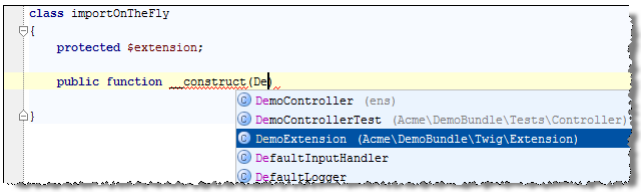
- Have PhpStorm automatically complete the fully qualified class name, including the namespace the class is defined in. This will result in littering your code.
- Have PhpStorm automatically complete the short class name and import the namespace the class is defined in.
- Import the namespace manually using a quick fix.

In the PHP context, a namespace is imported by inserting a `use` statement.

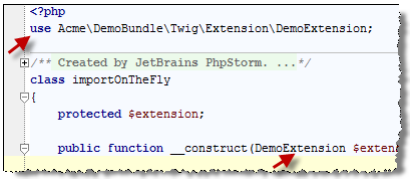
Completing a short class name and importing a PHP namespace on-the-fly

To compile a short class name and import a PHP namespace, follow these steps:

1. To enable on-the-fly namespace import, open the PhpStorm settings , and then click Auto Import under the Editor node. In the Editor: Auto Import page that opens, configure automatic namespace import in the PHP section.
   - To have PhpStorm automatically import PHP namespaces, add use statements, and complete short class names on the fly when you are typing in a class or file that belongs to a certain namespace, select the Enable auto-import in namespace scope check box. This check box is selected by default.
   - To have automatic namespace import applied when you are typing in a file that does not belong to any specific namespace, select the Enable auto-import in file scope check box.
2. Open the desired file for editing and start typing the short name of a class.
3. From the suggested variants for completion , select the relevant class name:



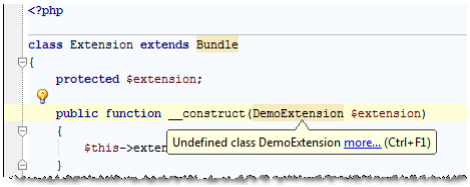
PhpStorm completes the short class name and inserts a `use` statement with the namespace where the selected class is defined:



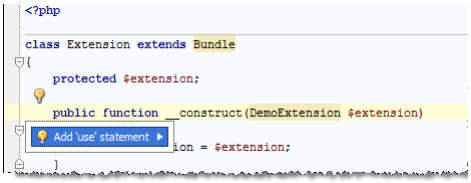
Importing a PHP namespace using a quick fix

To import a PHP class using a quick fix, follow these steps:

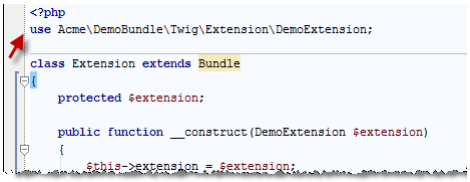
1. Open the desired file for editing and reference a PHP class. If the referenced class is not bound, the following prompt appears:



2. Press `Alt+Enter`. PhpStorm suggests to import the namespace where it has detected the declaration of the class:



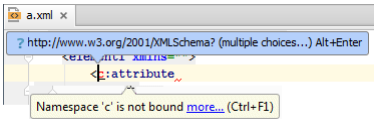
3. Press Enter. PhpStorm inserts a namespace declaration statement (`use` statement):



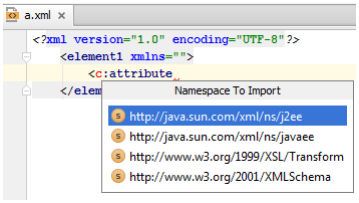
Importing an XML namespace

To import an XML namespace, follow these steps:

1. Open the desired file for editing, and start typing a tag. If a namespace is not bound, the following prompt appears:



2. Press `Alt+Enter`. If there are multiple choices, select the desired namespace from the list.

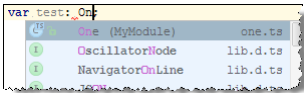


PhpStorm creates a namespace declaration.

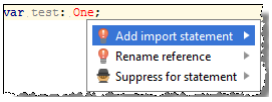
Importing TypeScript symbols

In the TypeScript context, PhpStorm can generate `import` statements for modules, classes, and any other symbol that can be exported and called as a type. Open the desired file in the editor and do one of the following:

- Start typing the short name of a symbol. From the suggested variants for completion, select the relevant symbol name:



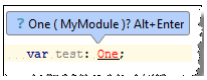
- Position the cursor at the unresolved symbol, which is displayed in red, and press `Alt+Enter`:



On the context menu, select Add import statement and press `Enter`.

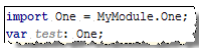
- Configure PhpStorm to show a pop-up every time you hover the mouse pointer over an unresolved reference which required import:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Auto Import under General.
  2. On the Auto Import page that opens, select the Show import pop-up check box in the TypeScript area.

  Every time you hover the mouse pointer over an unresolved symbol, PhpStorm will display the following pop-up message:



Press `Alt+Enter` to have an import statement generated and inserted automatically.

In either case, PhpStorm inserts an `import` statement:



Optimizing Imports

On this page:

- Introduction
- Optimizing imports in project
- Optimizing imports in the current file

Introduction

Sooner or later, some of the imported classes become redundant to the code.

Typically, you have to stop what you are doing, scroll to the head of the file, find the unused imports, and remove them. It is rather easy to forget to remove imports when you remove usages.

PhpStorm provides the Optimize Imports feature, which enables you, whenever it is convenient, to remove unused imports from your current file, or from all files in the current directory at once. This helps you avoid unused, excessive and duplicating imports in your project.

One can remove unused import statements in the entire project or in the current file only.

Optimizing imports in project

To optimize imports in the entire project, follow these steps:

1. Place the caret at the Project tool window and do one of the following:
   - On the main menu, choose Code | Optimize Imports.
   - Press `Ctrl+Alt+O`.

   The Optimize Imports dialog box opens.
2. If your project is under version control, the option Only VCS changed files is enabled. Select or clear this option as required.
3. Click Run.

Optimizing imports in the current file

One way of dealing with unused import is to use the quick-fix that appears when you set the caret at the highlighted unused import . However, you can optimize imports in a larger scope as described below.

To optimize imports in the currently opened file, do one of the following:

- On the main menu, choose Code | Optimize Imports.
- Press `Ctrl+Alt+O`.
- Place the caret at the import statements and press `Alt+Enter`, or just click to show the list of suggested intention actions, and then choose Optimize imports.

Code Inspection

In this section:

- Code Inspection
  - Code analysis basics
  - Inspection profiles
  - Inspection severity
  - Inspection scope
  - Examples of code inspections
    - Locating dead code
    - Highlighting unused declarations
    - Unresolved JavaScript function or method
    - Examples of PHP Code Inspections
- Accessing Inspection Settings
- Analyzing Inspection Results
- Configuring Inspection Severities
- Changing Highlighting Level for the Current File
- Customizing Profiles
- Disabling and Enabling Inspections
- Changing the Order of Scopes
- Exporting Inspection Results
- Resolving Problems
- Suppressing Inspections
- Running Inspections
- Running Inspection by Name
- Running Inspections Offline
- Applying Quickfixes Automatically

Code analysis basics

PhpStorm features robust, fast, and flexible static code analysis. It detects compiler and runtime errors, suggests corrections and improvements before you even compile.

PhpStorm performs code analysis by applying inspections to your code. Numerous code inspections exist for PHP and for the other supported languages.

The inspections detect not only compiling errors, but also different code inefficiencies. Whenever you have some unreachable code, unused code, non-localized string, unresolved method, memory leaks or even spelling problems – you'll find it very quickly.

PhpStorm's code analysis is flexibly configurable. You can enable/disable each code inspection and change its severity , create profiles with custom sets of inspections, apply inspections differently in different scopes , suppress inspections in specific pieces of code, and more.

The analysis can be performed in several ways:

- By default, PhpStorm analyses all open files and highlights all detected code issues right in the editor. On the right side of the editor you can see the analysis status of the whole file (the icon in the top-right corner).
  When an error is detected, this icon is ; in case of a warning, it is ; if everything is correct, the icon is .
- Alternatively, you can run code analysis in a bulk mode for the specified scope, which can be as large as the whole project.
- If necessary, you can apply a single code inspection in a specific scope.

For the majority of the detected code issues, PhpStorm provides quick fix suggestions . You can quickly review errors in a file by navigating from one highlighted line to another by pressing `F2` `Shift+F2` .

For more information and procedural descriptions, see Configuring Inspection Severities .

Inspection profiles

When you inspect your code, you can tell PhpStorm which types of problems you would like to search for and get reports about. Such configurations can be preserved as inspection profiles.

An inspection profile defines the types of problems to be sought for, i.e. which code inspections are enabled/disabled and the severity of these inspections. Profiles are configurable in the Inspections settings page.

To set the current inspection profile (the one that is used for the on-the-fly code analysis in the editor), simply select it in the Inspections settings page and apply changes. When you perform code analysis or execute a single inspection , you can specify which profile to use for each run.

Inspection profiles can be applicable for the entire IDE or for a specific project:

- Project profiles are shared and accessible for the team members via VCS. They are stored in the project directory: `<project>/.idea/inspectionProfiles` .
- IDE profiles are intended for personal use only and are stored locally in XML files under the `USER_HOME/.<PhpStorm version>/config/inspection` directory.

PhpStorm comes with the following pre-defined inspection profiles:

- Default: This local (IDE level) profile is intended for personal use, applies to all projects, and is stored locally in the `Default.xml` file under the `USER_HOME/.<PhpStorm version>/config/inspection` directory.
- Project Default: when a new project is created, the Project Default profile is copied from the settings of a template project . This profile is shared and applies to the current project.

  After a project is created, any modifications to the project default profile will pass unnoticed to any other projects.

When the settings of the Project Default profile are modified in the Template Project settings , the changed profile will apply to all newly created projects, but the existing projects will not be affected as they already have a copy of this profile.

Project Default profile is stored in the `Project_Default.xml` file located in the `<project>/.idea/inspectionProfiles` directory.

One can have as many inspection profiles as required. There are two ways of creating new profiles: you can add a new profile as a copy of the Project Default profile or copy the currently selected profile. The newly created profiles are stored in XML files, located depending on the type of the base profile .

The `<profile_name>.xml` files representing inspection profiles appear whenever some changes to the profiles are done and applied. The files only store differences against the default profile.

Refer to the section Customizing Profiles for details.

Inspection severity

Inspection severity indicates how seriously the code issues detected by the inspection impact the project and determines how the detected issues should be highlighted in the editor. By default, each inspection has one of the following severity levels:

- Server problem
- Typo
- Info
- Weak Warning
- Warning
- Error

You can increase or decrease the severity level of each inspection. That is, you can force PhpStorm to display some warnings as errors or weak warnings. In a similar way, what is initially considered a weak warning can be displayed as a warning or error, or just as info.

You can also configure the color and font style used to highlight each severity level. Besides, you can create custom severity levels and set them for specific inspections.

If necessary, you can set different severity levels for the same inspection in different scopes .

All modifications to inspections mentioned above are saved in the inspection profile currently selected in the inspection settings and apply when this profile is used.

Inspection scope

By default, all enabled code inspections apply to all project files. If necessary, you can configure each code inspection (enable/disable , change its severity level and options) individually for different scopes . Such configurations, like any other inspection settings, are saved and applied as part of a specific profile .

There may be complicated cases when an inspection has different configurations associated with different scopes. When such inspection is executed in a file belonging to some or all of these scopes, the settings of the highest priority scope-specific configuration are applied. The priorities are defined by the relative position of the inspection's scope-specific configuration in inspection settings : the uppermost configuration has the highest priority. The Everywhere else configuration always has the lowest priority.

For more information and procedural descriptions, see Configuring Inspection for Different Scopes .

Examples of code inspections

In the Inspections page, all inspections are grouped into categories. The most common tasks covered by code analysis are:

- Finding probable bugs.
- Locating dead code.
- Detecting performance issues.
- Improving code structure and maintainability.
- Conforming to coding guidelines and standards.
- Conforming to specifications.

Locating dead code

PhpStorm highlights in the editor pieces of so-called *dead* code. This is the code that is never executed during the application runtime. Perhaps, you don't even need this part of code in your project. Depending on situation, such code may be treated as a bug or as a redundancy. Anyway it decreases the application performance and complicates the maintenance process. Here is an example.

So-called *constant conditions* - conditions that are never met or are always *true*, for example. In this case the responsible code is not reachable and actually is a *dead* code.

```
attribute = parseAttribute(isempty, asp, php);

if (attribute == null) {
    ...
    return;
}
value = parseValue(attribute, false, isempty, delim);

if (attribute != null) {
    ... Condition 'attribute != null' is always 'true'.
}
else {
    av = new AttVal( null, null, null, null,
                     0, attribute, value );
    Report.attrError(this, this.token, value,
                     Report.BAD_ATTRIBUTE_VALUE);
}
```

PhpStorm highlights the if condition as it's always true. So the part of code surrounded with else is actually a dead code because it is never executed.

Highlighting unused declarations

PhpStorm is also able to instantly highlight Java classes, methods and fields which are unused across the entire project via Unused declarations inspection. All sorts of Java EE `@Inject` annotations, test code entry points and other implicit dependencies configured in the Unused declarations inspection are deeply respected.

Unresolved JavaScript function or method

This inspection detects references to undefined JavaScript functions or methods.

```
if (this.getValue())
Create Method 'getValue'
Rename Reference
```

Examples of PHP Code Inspections

- Unresolved Include

  This inspection detects attempts to include not actually existing files and suggests two quick fixes: to create a file with the specified name or use a PHPDOC annotation.

```
include 'notexist.php';
Create File notexist.php
Resolve using PHPDOC annotation
Inject Language
```

- Dynamic method is called as static

  This inspection checks whether a call to a static function is actually applied to a static function.

```
class Animal{
    public function do_something($direction = 0){
    }
}
Animal::do_something();
Inspection 'Dynamic method called as static' options
```

  The function `do_something()` is called as static while actually it is dynamic.

- Unimplemented abstract method in class

  This inspection checks whether classes inherited from abstract superclasses are either explicitly declared as abstract or the functions inherited from the superclass are implemented.

```
abstract class AbstractClass{
    abstract protected function getValue();
}

class ConcreteClass extends AbstractClass{
Inspection 'Unimplemented abstract methods in class' options

$w = new AbstractClass();
```

  The class `ConcreteClass` is inherited from an abstract class `AbstractClass` and has not been explicitly declared as abstract. At the same time, the function `GetValue()`, which is inherited from `AbstractClass`, has not been implemented.

- Parameter type

  PHP variables do not have types, therefore basically parameter types are not specified in definitions of functions. However, if the type of a parameter is defined explicitly, the function should be called with parameters of the appropriate type.

```
class Animal{
    public function do_something($direction = 0){
    }
}
$obj = new Animal();
$obj->do_something("run");
Inject Language
Inspection 'Parameter type' options
```

  The function `do_something` has the parameter of the type `integer` but is called with a `string`.

- Undefined class constant

  This inspection detects references to constants that are not actually defined in the specified class.

```
class Animal{
    public function do_something($direction = 0){
    }
}
$var = Animal::NotExistingConst;
Add constant declarat...
```

  The constant `NotExistingConst` is referenced to as a constant of the class `Animal`, while actually it is not defined in this class.

- Undefined constant inspection

  This inspection detects references to constants that are not actually defined anywhere within the inspection scope.

```
$var = UndefinedConst;
Inspection 'Undefined constant' options
```

  The referenced constant `UndefinedConst` is not defined anywhere within the inspection scope.

- Undefined class

  This inspection detects references to classes that are not actually defined anywhere within the inspection scope.

```
$var = new NotExistingClass();
Inspection 'Undefinded Class' options
```

  The referenced class `NotExistingClass` is not defined.

- Undefined field

  This inspection detects references to fields of a class that are not actually defined in this class.

```
class Animal{
    public function do_something($direction = 0){
    }
}
$obj = new Animal();
$var = $obj->field;
Inspection 'Undefined field' options
```

The `$obj` variable is an instance of the class `Animal`. The declaration of the `$var` contains a reference to the `field` field of the class `Animal`, which is not defined on this class.

In the PHP context, the Undefined field and Undefined method inspections may erroneously report severe problems when actually no problems take place. This happens when you attempt to access a property or to assign a value to a property that is not explicitly defined while the referenced class contains the `_get()` or `_set()` magic methods. No error should be reported because these methods are invoked every time an undefined property is referenced, however, PhpStorm still treats them as errors or warnings, depending on the severity you have specified for the inspection in general.

To suppress reporting Undefined method errors in such cases, re-configure the inspection severity. To do that, open the Inspections page of the Settings dialog box, click the inspection name in the list and select the Downgrade severity if __magic methods are present in class check box in the Options area. After that undefined properties in such cases will be indicated one step lower than specified for inspections in general, by default, Info instead of Warning.

To suppress irrelevant reporting of Undefined field errors, clear the Notify about access to a field via magic method and Notify about PHP dynamic field declaration check boxes. When the check boxes are selected, PhpStorm reports errors even if the class contains the `__get()` and `__set()` magic methods.

- Call of undefined function

  This inspection detects references to functions that are not defined anywhere within the inspection scope.

```
class Animal{
    public function do_something($direction = 0){
    }
}

$obj = new Animal();
$var = undefined_function($obj);
Inspection 'Undefined function' options
```

  The called function `undefined_function()` is not defined anywhere within the inspection scope.

- Undefined variable

  This inspection detects references to variables that are not declared and initialized anywhere within the inspection scope. PHP does not require that each variable should be declared and initialized. PHP can initialize such variable on the fly and assign it the zero value. However, this inspection allows you to detect discrepancies of this kind.

  The Undefined variable inspection can be configured through the check boxes on the Inspections page of the Settings dialog:

  - Enable inspection in global space: select this check box to run the inspection against variables outside functions/methods, classes, and namespaces, that is, in the global space.

```
<?php
/** Created by PhpStorm. ...*/

$y = $c;
Undefined variable 'c' more... (Ctrl+F1)
```

  - Report that variable might have not been defined: select this check box to have a warning displayed even if the definition of a variable is not definitely missing. Such situation may take place when a variable is used in several paths and some of them may never be reached, for example, in `if()` statements:

```
function test($r){
    if (($r==0) {
        $d=5;
    }
    return $d;
Variable 'd' might have not been defined more... (Ctrl+F1)
```

  - Ignore 'include' and 'require' statements Suppose the inspection scope contains an `include` or `require` statement. If this check box is cleared, PhpStorm treats such variable as defined in the classes referenced through these statements and no error is reported. If this check box is selected, an Undefined variable error is reported.

```
$var = $undefined;
Inspection 'Undefined variable' options
```

Accessing Inspection Settings

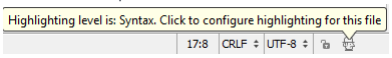
Inspections and inspection profiles are editable in the Inspections settings page. PhpStorm provides several ways to gain access to inspection settings.

To access inspections and profiles settings, do one of the following

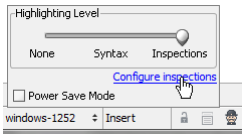
- Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Inspections under Editor.

- On the main toolbar, click , then expand the Editor node, and click Inspections .

- Click the current profile icon in the Status bar



and then click the Configure inspection link.

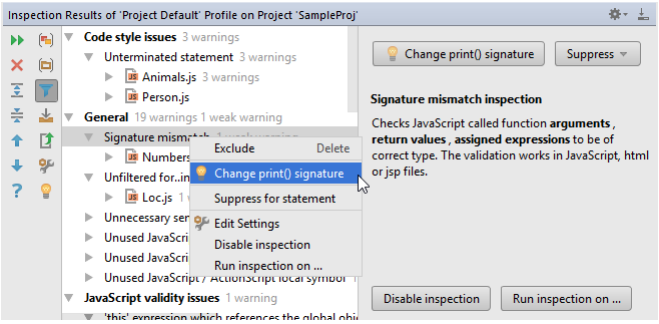


- In the editor, open the suggestion list, click the right arrow, and choose Edit inspection profile settings on the submenu.

- In in the Inspection tool window , click Edit Settings on the toolbar or use the corresponding context menu command.

Analyzing Inspection Results

After you perform code analysis or execute a single inspection , you can examine the results in the Inspection tool window .



Each run of the code analysis or a single inspection is displayed in a new tab in the tool window. The left part of the tab displays detected code issues groped by inspection groups, inspections and files. If necessary, you can change the default grouping. When you select a specific issue, its report appears in the right part.

In the Inspection tool window you can:

- Resolve issues using quick-fixes where available.
- Jump to the source of the problem to resolve it manually.
- Export inspection results to an XML or HTML file.
- Disable inspections to skip all corresponding issues nex time you ran code analysis.
- Suppress inspections for specific code issues.
- Access inspection settings to configure inspections.

Configuring Inspection Severities

On this page:

- Basics
- Changing severity of an inspection
- Changing severity of an inspection for different scopes
- Changing the highlighting style for a specific severity level
- Defining a custom severity level

Basics

Inspection severity indicates how seriously the code issues detected by the inspection impact the project and determines how the detected issues should be highlighted in the editor. By default, each inspection has one of the following severity levels:

- Server problem
- Typo
- Info
- Weak Warning
- Warning
- Error

You can increase or decrease the severity level of each inspection. That is, you can force PhpStorm to display some warnings as errors or weak warnings. In a similar way, what is initially considered a weak warning can be displayed as a warning or error, or just as info.

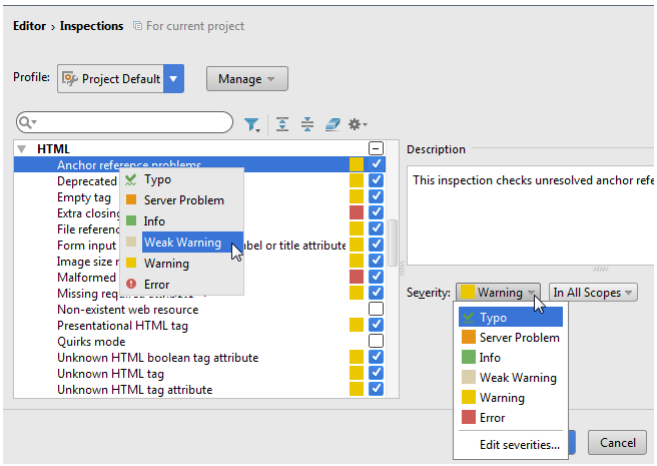
You can also configure the color and font style used to highlight each severity level. Besides, you can create custom severity levels and set them for specific inspections.

If necessary, you can set different severity levels for the same inspection in different scopes .

All modifications to inspections mentioned above are saved in the inspection profile currently selected in the inspection settings and apply when this profile is used.

Changing severity of an inspection

1. In the inspection settings , select the desired profile . The inspections associated with the profile are displayed in the tree view.
2. Select the desired inspection. If this inspection is disabled, select the check box next to it.
3. Select the desired severity from the context menu of the inspection or from the Severity selector on the right:



   Note that inspections, whose state is changed relative to the default values, and all their grouping nodes are highlighted with blue.

4. Apply the changes. The modified inspection will now have the new severity level when this profile is used.

In the PHP context, the Undefined field and Undefined method inspections may erroneously report severe problems when actually no problems take place. This happens when you attempt to access a property or to assign a value to a property that is not explicitly defined while the referenced class contains the `_get()` or `_set()` magic methods. No error should be reported because these methods are invoked every time an undefined property is referenced, however, PhpStorm still treats them as errors or warnings, depending on the severity you have specified for the inspection in general.

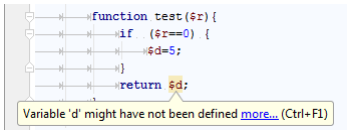
To suppress reporting Undefined method errors in such cases, re-configure the inspection severity . To do that, open the Inspections page of the Settings dialog box, click the inspection name in the list and select the Downgrade severity if __magic methods are present in class check box in the Options area. After that undefined properties in such cases will be indicated one step lower than specified for inspections in general, by default, Info instead of Warning.

To suppress irrelevant reporting of Undefined field errors, clear the Notify about access to a field via magic method and Notify about PHP dynamic field declaration check boxes. When the check boxes are selected, PhpStorm reports errors even if the class contains the `__get()` and `__set()` magic methods.

The Undefined variable inspection can be configured through the check boxes on the Inspections page of the Settings dialog:

- Enable inspection in global space: select this check box to run the inspection against variables outside functions/methods, classes, and namespaces, that is, in the global space .

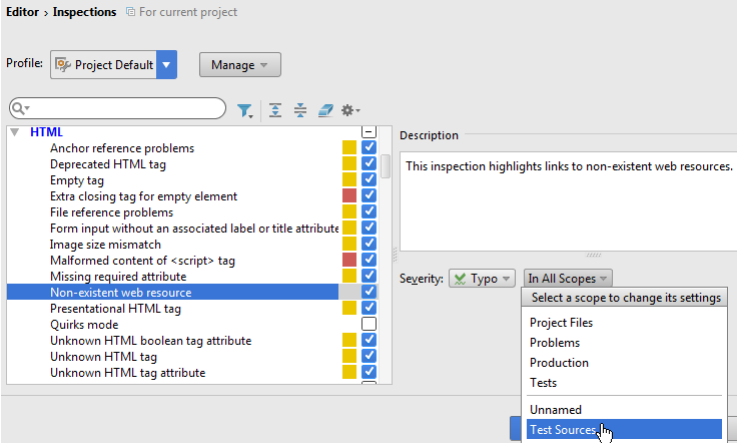
- Report that variable might have not been defined: select this check box to have a warning displayed even if the definition of a variable is not definitely missing. Such situation may take place when a variable is used in several paths and some of them may never be reached, for example, in `if()` statements:



- Ignore 'include' and 'require' statements Suppose the inspection scope contains an `include` or `require` statement. If this check box is cleared, PhpStorm treats such variable as defined in the classes referenced through these statements and no error is reported. If this check box is selected, an Undefined variable error is reported.

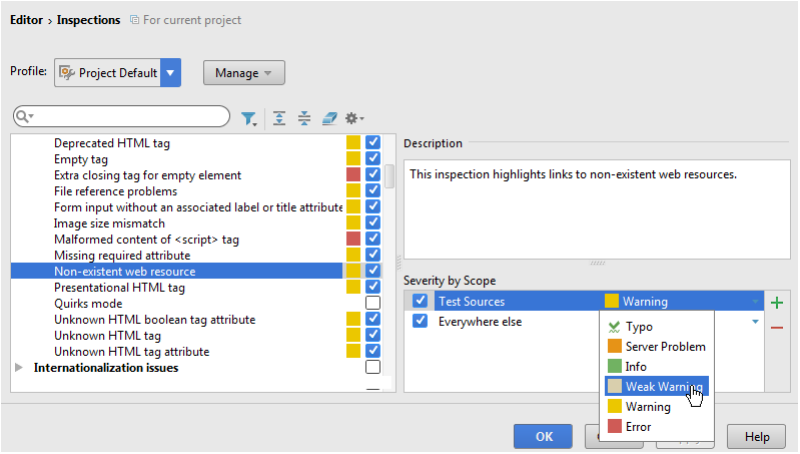
Changing severity of an inspection for different scopes

1. Choose the desired profile and inspection.
2. Click the drop-down list In All Scopes, and select the scope you want to change inspection severity in:



PhpStorm shows severities for two scopes: for the selected one and Everywhere else

3. Click severity drop-down list for the selected scope and choose the appropriate severity level from the drop-down list:



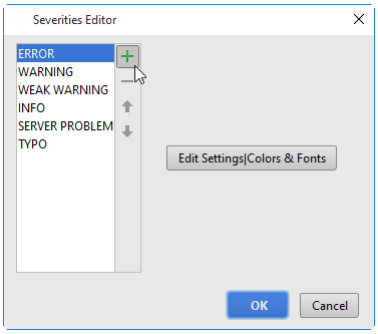
Changing the highlighting style for a specific severity level

1. Do one of the following:
   - In the Settings/Preferences dialog, select Editor | Colors & Fonts -> General, and then select the style corresponding to the desired severity level.
   - In the inspection settings , select the desired inspection and choose Edit severities from the Severity selector.
     Next, in the Severities Editor dialog box that opens, select the desired severity level and click Editor | Colors & Fonts.

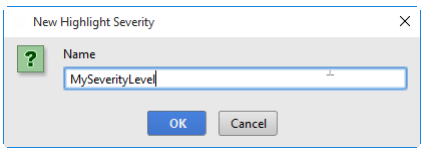
   Either way you will see the styles associated with this severity in the Colors and Fonts settings page.
2. Configure the color and font styles as necessary and apply changes. The detected issues with the corresponding severity will now be highlighted in the editor with the modified style when the current profile is used.

Defining a custom severity level

1. In the inspection settings , select the desired inspection and choose Edit severities from the Severity selector.
2. In the Severities Editor dialog box that opens, click +:



3. Type the name for the new severity in the New Highlight Severity dialog box.



The custom severity is added to the list of severities.

4. Specify color and font settings for the new severity using the controls to the right of the list of severities.
5. Use the Up and Down buttons to change the priority of the new severity.
6. Apply the changes. The new severity level will now be available for all inspections within the current profile . You can assign it to specific inspections and get the corresponding code issues highlighted with the specified style in the editor.
   If necessary, you can remove the custom severity level later by selecting it in the Severities Editor dialog box and clicking .

   You cannot change priorities of the pre-defined severity levels, or remove them.

Changing Highlighting Level for the Current File

Use the Status bar to quickly re-configure highlighting for the file which is currently opened in the editor. With Hector, you can choose to highlight syntax problems, inspection issues, or turn off highlighting.

To change the highlighting level for the current file

1. Open the Highlighting level pop-up window by doing one of the following:
   - On the main menu, choose Code | Configure Current File Analysis.
   - Press `Ctrl+Shift+Alt+H`.
   - Click the Hector icon on the Status bar.
   - Right-click the code inspection indicator on top of the scroll bar.

2. Move the slider to one of the three available positions that define the highlighting level:
   - None: turn off problems highlighting in the editor.
   - Syntax: highlight syntax problems only.
   - Inspections: (default) highlight syntax problems and inspection issues.

Customizing Profiles

On this page:

- Introduction
- Customizing profiles
- Creating and copying profiles
- Deleting profiles
- Importing and exporting profiles
- Adding or editing the profile description

Introduction

An inspection profile defines the types of problems to be sought for, i.e. which code inspections are enabled/disabled and the severity of these inspections. Profiles are configurable in the Inspections settings page.

To set the current inspection profile (the one that is used for the on-the-fly code analysis in the editor), simply select it in the Inspections settings page and apply changes. When you perform code analysis or execute a single inspection , you can specify which profile to use for each run.

PhpStorm allows you to customize the existing inspection profiles (including default profiles), and create new ones. You can also share, import and export inspection profiles.

PhpStorm distinguishes between global and project profiles.

Global profiles

These profiles are saved in an application config directory (for example, `~/.PhpStormXXXX/config/inspection` on Linux) and are available for any user project.

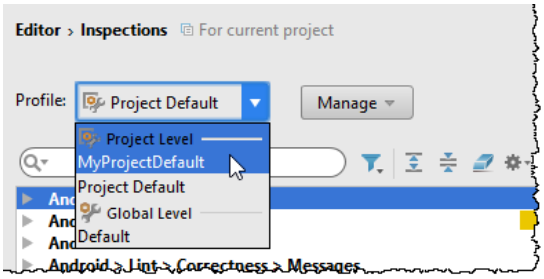
Project profiles

These profiles are saved in a particular project's `.idea` directory (for example, `$PROJECT_DIR/.idea/inspectionProfiles` on Linux).

Customizing profiles

To customize a profile

1. In the inspection settings , select the profile to be changed.



   Note that the default profiles are also editable.

2. Customize the desired inspections: enable or disable , change their severity and the other options, which can be different for the various inspections.
   Note that the selectors of severity and scopes, and the options section (if any) are only available for the enabled inspections.

3. To apply an inspection to a restricted set of files, associate it with the corresponding scopes .
   By default, all inspections apply to all the sources of the current project.

4. Apply changes.

Creating and copying profiles

To create or copy a profile

1. In the inspection settings , choose the profile that you want to copy.

2. Click Manage, and then choose Copy.

3. In the Profile field, type the name of the new profile. Press `Enter` to save the entry, or `Escape` to discard changes.

4. Apply changes. The new profile is created as the copy of the selected profile.

Deleting profiles

To delete a profile

1. In the inspection settings , select the profile to be deleted from the list of profiles.

2. Click Manage, and then choose Delete.

3. Apply changes.

Importing and exporting profiles

To import a profile

1. In the inspection settings , click Manage, and then choose Import.
2. In the Select Path dialog box, locate the desired `xml` file, and click OK.
3. Apply changes.

To export a profile

1. In the inspection settings , select the profile that you want to export.
2. Click Manage, and then choose Export.
3. In the Select Path dialog box, locate the desired directory, where the `xml` file of the current inspection profile will be saved, and click OK.
4. Apply changes.

Adding or editing the profile description

To add profile description

1. In the inspection settings , select the profile that you want to add a description for.
2. Click Manage, and then choose Add description. The field for entering description text appears.
3. Type the desired description, and press `Enter` to save the entry, or `Escape` to discard changes.

When a description of the profile already exists, the command changes to Edit Description.

Disabling and Enabling Inspections

On this page:

- Introduction
- Disabling or enabling inspections

Introduction

If you think that some inspections report about the problems that you are not interested in, you can disable such inspections. Note that when you disable an inspection, it is disabled in the current inspection profile ; in all other profiles, it remains enabled.

There are several ways to disable/enable inspections:

- Using the Inspections page in the Settings/Preferences dialog - this is the main interface for managing inspections; here you can see at once, which inspections are enabled or disabled in all inspection profiles.
- Using the intention actions - this is the way to disable a highlighted code issues right in the editor.
- In the Inspection tool window - this is a quick way to disable uninteresting issues when analyzing inspection results . Note that here you can only disable inspections.

Note Note the difference between disabling and suppressing code inspections:

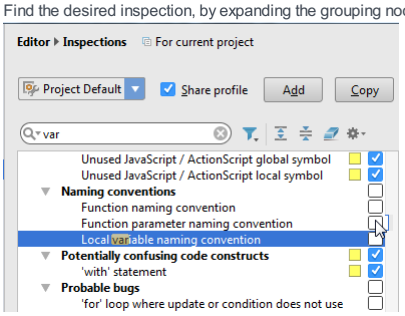
When suppressing an inspection, PhpStorm inserts a special comment that tells the code analysis engine to ignore the issues found by this inspection in the specific piece of code.

When disabling an inspection, you just turn it off so the code analysis engine just ignores the code issues found by this inspection.

Disabling or enabling inspections

To disable or enable an inspection in the Settings/Preferences dialog

1. Find the desired inspection, by expanding the grouping nodes or using the search field.



2. Use the check box next to the inspection to disable or enable it.
3. Apply the changes and close the dialog box.

To disable an inspection for highlighted issue in the editor

- When you disable inspections this way, they are disabled for the current inspection profile .
- To re-enable inspections disabled this way, use the main procedure described above.

1. Set the caret at a highlighted issue.
2. Click the bulb icon or press `Alt+Enter` to reveal the inspection alert and suggestion list.
3. Select the inspection to be disabled, then click right arrow button or just press the right arrow key.
4. On the submenu, click Disable <inspection name>.

To disable inspections from the Inspection results report

- When you disable inspections this way, they are disabled for the inspection profile that was used for running inspections . You can see it in the header of the Inspection window's tab.
- To re-enable inspections disabled this way, use the main procedure described above.

1. In the Inspection tool window , right-click the inspection you want to disable.
2. On the context menu, choose Disable inspection.
3. Press the filter button to hide the disabled inspection alerts.

Configuring Inspection for Different Scopes

On this page:

– Basics

– Defining the order of scopes

Basics

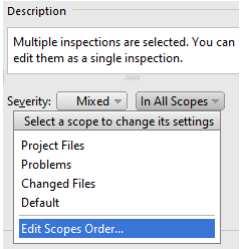
By default, all enabled code inspections apply to all project files. If necessary, you can configure each code inspection (enable/disable , change its severity level and options) individually for different scopes . Such configurations, like any other inspection settings, are saved and applied as part of a specific profile .

There may be complicated cases when an inspection has different configurations associated with different scopes. When such inspection is executed in a file belonging to some or all of these scopes, the settings of the highest priority scope-specific configuration are applied. The priorities are defined by the relative position of the inspection's scope-specific configuration in inspection settings : the uppermost configuration has the highest priority. The Everywhere else configuration always has the lowest priority.

Defining the order of scopes

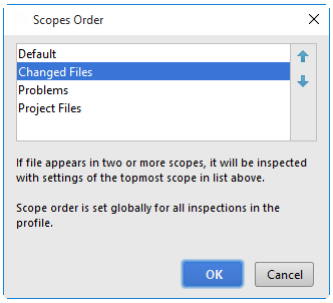
To define the order of scopes, follow these steps

1. In the Inspections page of the Settings/Preferences dialog, click the button In All Scopes:



2. Choose Edit Scopes Order... from the scopes drop-down list.

3. In the Scopes Order dialog box that opens, select the desired scope, and click the up and down arrows:



Exporting Inspection Results

After you perform code analysis or execute a single inspection , you can save the inspection results for further examination or for sharing with colleagues. PhpStorm enables you to export inspection results to the HTML or XML format.

To export inspection results

1. On the toolbar of the Inspection tool window, click the Export button .
2. From the Export To context menu, select the target format. The available options are HTML and XML.
3. In the dialog that opens , specify the target directory to store the inspection results in.

Resolving Problems

On this page:

- Introduction
- Fixing problems

Introduction

By default, PhpStorm analyses the code in all open files and highlights the detected code issues. You can fix most of the issues immediately by applying quick-fixes .

If you perform code analysis or execute a single inspection in a larger scope of source files, PhpStorm displays the detected code issues in the Inspection tool window. When you select a specific issue in this window, its report is shown in the right part of the window.

If there are available fixes to the issue, PhpStorm notifies it in the following ways:

- The Apply a Quickfix button becomes available on the Inspection's window toolbar.
- The available fixes are shown in the optional Problem resolution field of the report.
- The available fixes are shown in the context menu of the issue.

If there are no available fixes, the only option is to fix the issue manually.

Please note the following:

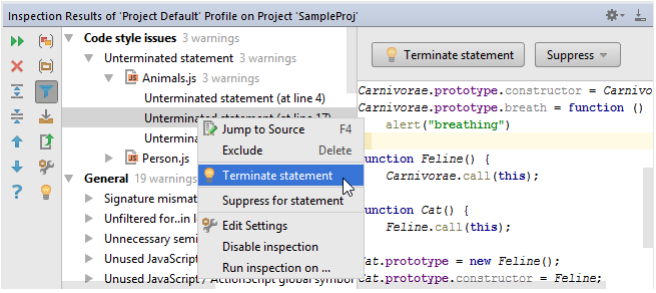
- To display the source code of an issue in the editor, when it is selected in the Inspections tool window, toggle the Autoscroll to Source button.
- If you find that some of the reported issues are minor or not helpful to you, you can ignore them either by disabling the corresponding inspection or by suppressing it in a specific piece of code.

Fixing problems

To fix a problem reported by code inspection

In the Inspections tool window, select the code issue you are interested in and do one of the following:

- If PhpStorm suggests any fixes to the issue as described above, you can use one of them to fix the problem immediately.



- If no resolutions are suggested, use the Jump to source command in the context menu and fix the problem manually.

Suppressing Inspections

On this page:

- Introduction
- Suppressing inspections in the editor
- Suppressing inspections from the Inspection tool window

Introduction

For some reasons, you may want to partly disable a specific inspection, i.e. ignore some code issues while continuing to detect the other issues with this inspection.

For example, PhpStorm considers some code to be "dead", and you can see that it is true. The inspection is helpful and you do not want to disable it. However, you may want to use this code later and do not want it to be highlighted in the editor or appear in the issue reports.

To do so, PhpStorm allows you to suppress certain inspections for a specific statement, function/method, tag or file. You can do it either in the editor , using the suggestion list or in the Inspection tool window when analysing inspection results.

Let's summarize the difference between suppressing and disabling code inspections:

When suppressing an inspection, PhpStorm inserts a special comment that tells the code analysis engine to ignore the issues found by this inspection in the specific piece of code.

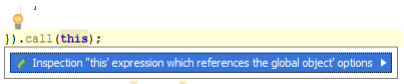
When disabling an inspection, you just turn it off so the code analysis engine just ignores the code issues found by this inspection.

Some code inspections (e.g. those detecting errors) cannot be suppressed.

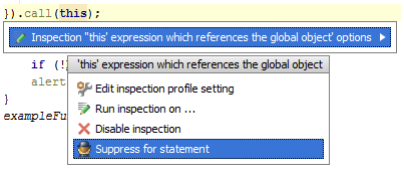
Suppressing inspections in the editor

To suppress an inspection in the editor

1. Set the cursor to the highlighted code issue in the editor.
2. Press `Alt+Enter`, or click the light bulb icon to expand the suggestion list.



3. Depending on the issue, you will see either quick-fixes related to the inspection or the Inspection "<inspection name>" options item.
4. Use the up/down arrow keys to select this item and then press the right arrow key or just click the right arrow ▸ next to this item.
   Pressing the left arrow key, or `Escape` hides the suggestion list.
5. In the inspection options list, select the desired suppress action:

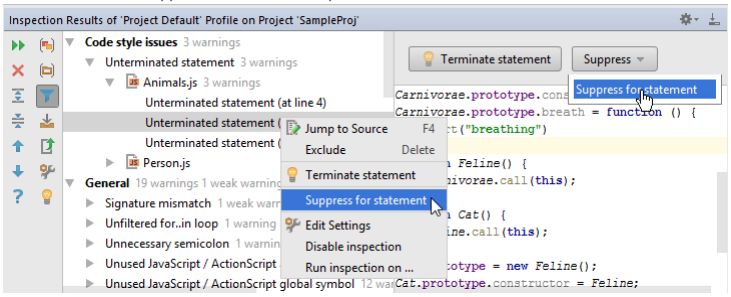


The inspection will be suppressed with special comments in the corresponding piece of code.

Suppressing inspections from the Inspection tool window

To suppress inspections from the Inspection tool window

1. After running code analysis , select a code issue, for which you want to suppress the inspection, in the Inspection tool window.
2. Click the button Suppress and choose the scope of sustension, or just right-click the selected inspection.
3. Choose the desired suppress action. For example:



The inspection will be suppressed with special comments in the corresponding piece of code.

Running Inspections

Although code analysis is performed on-the-fly in all open files, you may want to run it for the whole project or a custom scope and examine the results in a friendly view.

Note Inspecting code in a large scope, for example, the whole project, can be time consuming. For large projects, it is recommended to create a number of smaller scopes and run analysis in each of them separately.

– If you do not want to have some code issues in the analysis report, you can disable some inspections .

To run a code inspection

1. Open the desired file in the editor. Alternatively select files or directories in the Project tool window . For multiple selection, click the items holding down the Ctrl / ⌘ key. The initial inspection scope will be confined to the opened file or the selection.

2. On the main menu, choose Code | Inspect Code. The Specify Inspection Scope dialog box opens.

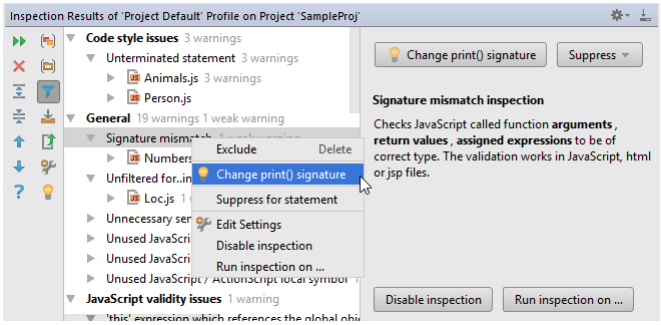
3. In the Inspection scope area, specify which files should be inspected.
   – To have the source code of the entire project inspected, select the Whole Project option.
   – If you are using version control integration , you can choose to only inspect uncommitted files.
   – To run an inspection for the currently opened file, or the file(s)/folder(s) selected in the Project view, select the File/Module <name> option.
   – To apply inspect code in a specific scope , select the Custom scope option, then choose the desired scope from the drop-down list or click the Browse button ... and configure a new scope in the Scopes dialog box.

4. To have test source files inspected too, select the Include test sources check box.

5. Specify the inspection profile to apply. Do one of the following:
   – Select one of the existing profiles from the Inspection Profile drop-down list.
   – Click the Browse button ... and configure a new profile in the Inspections dialog box.

6. Click OK to run code analysis.

7. Examine results in the Inspection tool window.



It is also possible to run inspections offline, without starting the IDE. Follow the procedure described in the section Working with PhpStorm Features from Command Line .

Running Inspection by Name

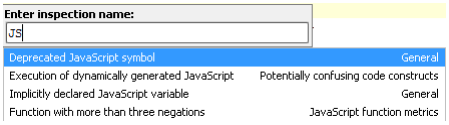
Rather than running all enabled inspections, PhpStorm makes it possible to exactly specify the desired inspection by its name and run it to inspect the code in the specific scope.

To run a code inspection by name

1. On the main menu, choose Code | Run Inspection by Name, or press `Ctrl+Shift+Alt+I`.

2. In the pop-up frame that opens, start typing the inspection name. As you type, the suggestion list shrinks to show the matching inspection only.

   Tip Use CamelHumps to match camel case words and white spaces to match initial letters of the words.

   The Specify Inspection Scope dialog box opens.

3. In the Inspection scope area, specify which files should be inspected.
   - To have the source code of the entire project inspected, select the Whole Project option.
   - If you are using version control integration , you can choose to only inspect uncommitted files.
   - To run an inspection for the currently opened file, or the file(s)/folder(s) selected in the Project view, select the File/Module <name> option.
   - To apply inspect code in a specific scope , select the Custom scope option, then choose the desired scope from the drop-down list or click the Browse button ... and configure a new scope in the Scopes dialog box.

4. To have test source files inspected too, select the Include test sources check box.

5. To apply inspection only in files matching the specific mask, select the File Masks(s) and specify the file mask. Use comma to separate multiple file masks.

6. Click OK to run the inspection.

7. Examine results in the Inspection tool window.

Running Inspections Offline

On this page:

- Basics
- Launching a code inspection from the command line
  - Examples
    - Windows
    - OS X
- Viewing the results of an offline inspection

Basics

In addition to running code inspections from the main menu, or from the context menus of the Project tool window , you can also launch the inspector from the command line, without actually running PhpStorm.

This way you can perform regular code inspections as a part of your development process, which is especially important for large projects. Inspection results are stored in the XML format.

Launching a code inspection from the command line

To launch a code inspection from the command line

- Specify the following command line arguments:
  - Path to the launcher: specify the full path to one of the following launchers (which reside under the `bin` directory of your PhpStorm installation):
    - For Windows: `inspect.bat`
    - For UNIX and OS X: `inspect.sh`
  - Project file path is the full path to the directory that contains the project to be inspected.
  - Inspection profile path is the full path to the profile, against which the project should be inspected. The inspection profiles are stored under `USER_HOME\.PhpStormXX\config\inspection`
  - Output path is the full path to an existing directory where the report will be stored.
  - Options. You can specify:
    - The directory to be inspected `-d <full path to the subdirectory>`
    - The verbosity level of output `-vX` , where X is 0 for quiet, 1 for noisy and 2 for extra noisy.

Warning! Please note that you have to specify full paths. Relative paths are not accepted!

Examples

Windows

```
C:\Program Files (x86)\JetBrains\<PhpStorm home>\bin\inspect.bat
E:\SampleProjects\MetersToInchesConverter E:\Work\MyProject\.idea\inspectionProfiles\Project_Default.xml
E:\Work\MyProject\inspection-results-dir -v2 -d E:\SampleProjects\MetersToInchesConverter\subdirectory
```

OS X

```
/Applications/PhpStorm.app/Contents/bin/inspect.sh ~/PhpStormProjects/MyTestProject
~/Library/Preferences/PhpStorm<version>XX/inspection/Default.xml ~/PhpStormProjects/MyTestProject/results-dir -v2
```

Viewing the results of an offline inspection

If you have performed an offline inspection and exported the inspection results to a directory in the XML format you can always download and view these results.

To view the results of an offline inspection, follow these steps

1. Open the project against which the inspection was performed.
2. On the main menu, choose Code | View Offline Inspection Results.
3. In the Select Path dialog box that opens, navigate to the directory that contains inspection results in XML format.
4. Click OK. Inspection results display in the Offline View tab in the Inspection tool window .

Tip Alternatively, you can open the relevant XML file in PhpStorm or in any other text processor without opening the inspected project.

Applying Quickfixes Automatically

On this page:

– Introduction

– Applying quickfixes automatically

– Applying quickfixes on commit to VCS

Introduction

When you run an inspection against your code, the Inspection Tool Window is displayed showing the results of code analysis. You can then examine the issues that have been detected and apply the suggested quickfixes .

However, you may want to streamline this process by automatically applying the quickfixes from your inspection profile to the selected scope, without having to examine the results and implement each separate fix individually.

Applying quickfixes automatically

To apply quickfixes automatically

1. From the main menu, select Code | Code Cleanup.
2. In the Specify Code Cleanup Scope dialog that opens, select ths scope to which you want the inspection profile to be applied.
3. Select the inspection profile from the drop-down list, or click the button to configure a new profile in the Inspections dialog that opens. You can also click the button to check which fixes will be applied.
4. Click OK to launch the inspection.

PhpStorm will perform code analysis and will automatically apply quickfixes from the selected inspection profile to all issues that are detected.

Alternatively, you can place the caret at an error in the source code that corresponds to a quickfix, click the red bulb (quick-fix suggested) icon that appears on the left, and select Code Cleanup from the drop-down menu. Code cleanup will be performed for the current file with the current inspection profile.

Applying quickfixes on commit to VCS

To apply quickfixes when committing changes to a VCS

1. From the main menu, select VCS | Commit Changes.
2. In the Commit Changes dialog that opens, select the files that you want to commit, and select the Cleanup code option in the Before Submit area.

Code cleanup will be performed for all selected files before they are committed to your version control system (the current project inspection profile will be applied).

Intention Actions

In this section:

- Intention Actions
  - Introduction
  - Intention action icons
  - Intention action types
- Applying Intention Actions
- Configuring Intention Actions
- Disabling Intention Actions

Introduction

PhpStorm helps you handle the situations when you use classes that haven't been imported, or methods that haven't been written etc., which can result in errors. When a possible problem is suspected, PhpStorm suggests a solution, and in certain cases can implement this solution (properly assign variables, create missing references and more). Besides syntax problems, PhpStorm recognizes code constructs that can be optimized or improved, and suggests appropriate intention actions, denoted with the special icons.

Intention action icons

Item	Icon	Description
Intention actions suggested		A yellow bulb indicates that PhpStorm just proposes to alter your code. It covers a range of situations from warning correction to suggestions for code improvement (like micro-refactorings).
Specific intention action		This sign appears in the suggestion list before each specific intention action. If an intention action alert is disabled, the sign turns to . Disabled intention action is still available and can be enabled again.
Quickfix suggested		A red bulb with an exclamation mark indicates that PhpStorm suggests a way to fix an error. It is related to Create from usage intentions and Quick fixes.
Disabled		Alert is disabled, but the intention action is still available and can be enabled again.

Intention action types

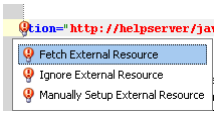
Find descriptions of specific intention actions on the Intentions page of the editor settings/preferences, where they are grouped according to the areas of their usage. Generally, intention actions can be divided into several categories, for example:

Create from usage

This type of intention action creates new code items: classes, methods, etc. They are smart enough to analyze your code and provide actions suitable for a particular case. The main concept behind this type is that you can begin using new things without declaring them first. You are not taken away from your current task for mundane minutiae like creating declarations, new files, etc. which PhpStorm handles while you keep focused.

Quick fixes

This type of intention action responds to common coding mistakes: using an improper access modifier, or an expression of the wrong type, or missing resources, etc. PhpStorm catches these kinds of problems as you type, and provides a quick way to fix them using Intentions Actions with appropriate suggestions for the error.



Micro-refactorings

These intention actions appear for code that is syntactically correct, but can be structurally improved by such things as:

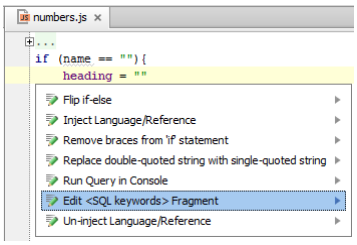
- Converting code constructs.

- Splitting declarations and assignments.

- Splitting or merging statements and tags, etc.

Edit <Injected Language> Fragment

For string literals that represent language injections , the Edit <Injected Language> Fragment intention action is available. You can use this intention action to open the corresponding code fragment in a separate editor.



Applying Intention Actions

PhpStorm displays an intention action alert, and it is your task to define which action (if any) should be performed.

To apply an intention action

1. Click the light bulb icon, or use the `Alt+Enter` keyboard shortcut to bring up the suggestion list.
2. Click the desired option, or use the arrow keys to select the option and press `Enter`.

Configuring Intention Actions

On this page:

– Introduction

– Configuring intention settings using the Settings/Preferences dialog

– Configuring intentions on-the-fly

Introduction

PhpStorm makes it possible to configure intention action settings either in the Intentions page of the Settings/Preferences dialog, or "on-the-fly".

By default, all available intention actions that ship with PhpStorm, are enabled. By changing the Intention settings, you can disable the actions that are not required for your current working environment.

Configuring intention settings using the Settings/Preferences dialog

To configure intention settings using the Settings/Preferences dialog

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Intentions under Editor.
2. In the Intentions page, clear the check boxes of the intention actions or action categories that you do not currently need.
   Selecting or clearing a category affects all intention actions in this category.
3. Apply changes and close the dialog.

Configuring intentions on-the-fly

To configure intention settings on-the-fly

1. In the editor, press `Alt+Enter` to reveal the inspection alert and suggestion list.
2. Select the action to be disabled, then click right arrow button or just press the right arrow key.
3. On the submenu, choose Disable <intention action name>.

Disabling Intention Actions

If an intention action is enabled, the alert shows automatically, when the caret rests on the problem-causing piece of code. You can disable alert for any type of intention actions, and show it by explicit invocation only.

Disabled intention actions are marked with the grey light bulb icon. Though disabled, such actions are still available and can be applied to the source code.

You can suppress intention actions related to inspections "on-the-fly", as described in the section Suppressing Inspections .

To disable an intention action alert

1. Click `Alt+Enter`, or click the light bulb icon to show the suggestion list.
2. Select the action to be disabled and click the right arrow button.
3. On the submenu, click Disable <intention action name>:

For the disabled intention action, the menu item changes to Enable <intention action name>.

Viewing Pages with Web Contents

With PhpStorm, you can perform two opposite operations:

- Preview the output of your Web application in the browser to check whether the pages are rendered correctly. PhpStorm can display a page preview in a browser of your choice or you can switch between several browsers. PhpStorm currently supports previews in the following browsers:
  - Mozilla Firefox
  - Internet Explorer
  - Safari
  - Chrome
  - Opera

  No preliminary steps are required to view HTML code. However, to preview PHP pages in the browser, you need to configure synchronization with a server first.
- View the HTML source code of a Web page in the PhpStorm editor.

In this section:

- Configuring Browsers
- Previewing Pages with Web Contents in a Browser
- Viewing HTML Source Code of a Web Page in the Editor

Configuring Browsers

On this page:

- Integrating browser installations with PhpStorm
- Choosing the default PhpStorm browser

Integrating browser installations with PhpStorm

To make it possible to launch a Web browser from PhpStorm, you need to integrate installations of Web browsers with PhpStorm, activate or deactivate launching Web browsers from PhpStorm and specify whether a browser will be launched by running its executable file or through the default system command .

PhpStorm is shipped with a predefined list of most popular browsers which you may like to install and use. The items are added to the list in advanced and are not based on the information on actually installed browsers. PhpStorm presumes that you install browsers according to a standard procedure. Based on this assumption, each browser in this predefined list is assigned an alias which stands for the path to its executable file, as PhpStorm supposes it to be. If in your actual browser installation the path to the executable file is different, you need to specify it explicitly as described below.

In addition to the predefined browsers, you can configure as many custom browser installations as you need using the controls on the toolbar. To create a list of Web browser that can be launched from PhpStorm:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Web Browsers under Tools.
2. The Web Browsers page that opens shows a predefined list of browsers, possibly extended with previously configured custom browser installations. Each browser is presented as a separate table row. The fields in each row show the name of the browser, the family to which the browser belongs, and the path to the browser's executable file or the predefined alias that stands for this path.
   - To activate an actually installed browser, select the Active check box next to its name. The browser will be added to the context menu of the Open in Browse menu item and its icon will be displayed in the Browsers pop-up toolbar.
     If the browser was installed according to a standard installation procedure, most likely the alias shown in the Path field points at the right location of the executable file. To specify the path explicitly, click ... in the Path field and choose the actual location of the executable file in the dialog box that opens.
   - To configure a custom browser installation, click + on the toolbar. In the new row that is added to list, specify the browser name, family and the path to its executable.
   - To change the order of browsers in the list, use the ↑ and ↓ buttons. The order of browsers in the list affect the order in which they will be shown on the context menu of the Open in Browse menu item.
   - To specify a custom profile for Firefox or a browser of the Chrome family, select the browser in question, click on the toolbar. Depending on the family of the selected browser, the Firefox Settings or Chrome Settings dialog box opens.
     - For Firefox, specify the path to the required `profiles.ini` file and choose the profile to use from the drop-down list. Learn more at Firefox browser profile .
     - For Chrome, select the Use custom profile directory check box and specify the location of the `chrome-user-data` folder where users' profiles are stored. Learn more about Chrome profiles at Multi-profiles .
   - To launch a browser of the Chrome with additional options, click on the toolbar and type the required keys in the Command Line Options text box of the Chrome Settings dialog box that opens. Learn more about Chrome command line options by opening chrome://flags in Chrome.
   - To remove a browser from the list, select the browser and click — on the toolbar. Note that only custom browser can be removed.

Choosing the default PhpStorm browser

When you want to preview your application output in the browser by choosing View | Open in Browser on the main menu or Open in Browser on the context menu of a file, you need to choose the browser to open the preview in. You can use a specific browser from the context menu or choose Default Browser. tell PhpStorm which browser you want to be used by default. This browser is called PhpStorm default browser.

PhpStorm also opens the PhpStorm default browser to render external resources.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Web Browsers under Tools. The Web Browsers page opens.
2. From the Default Browser drop-down list, choose the browser to use by default for previewing pages.
   - To use the default operating system browser, choose System default.
   - To use the browser on top of the list, choose First listed. Change the order or browsers using the ↑ and ↓ icons on the toolbar.
   - To use another browser as default, choose Custom path and specify the location of the executable file of the required browser. Type the path manually or use the Browse button ..., if necessary.
3. Specify the way to have the browsers launched.
   - To have a popup window with the enabled browsers appear in the HTML files, select the Show browser popup in the editor check box.
   - If the Show browser popup in the editor check box is cleared, previewing HTML files is available only through the View | Open in Browser command on the main menu or the Open in Browser command on the context menu of a file.

Previewing Pages with Web Contents in a Browser

You can preview a file with Web contents in a browser. This can be the PhpStorm default browser specified in the Web Browsers section of the IDE settings or the one of your choice.

No preliminary steps are required to view HTML code. However, to preview PHP pages in the browser, you need to configure synchronization with a server first.

To preview an opened file in a Web browser, do one of the following:

- On the main menu, choose View | Open in Browser. The current file opens in the default browser.
- With the editor tab having the focus, choose View | Preview file in on the main menu or press `Alt+F2`. Then select the desired browser from the pop-up menu:

- Hover your mouse pointer over the code to show the browser icons bar, and click the icon that indicates the desired browser:

Tip If you don't want to see the icons toolbar, or would like to see the icons of just those browsers you are interested in, clear the Active check boxes for the unnecessary browsers in the Web Browsers section of the IDE settings.

Viewing HTML Source Code of a Web Page in the Editor

You can open the HTML source code of any Web page in the PhpStorm editor.

No matter which programming language was originally used to develop a page (for example, XML or PHP), PhpStorm shows the resulting HTML code.

To open the HTML source code of a Web page in the editor

1. Choose File | Open URL.
2. In the Open URL dialog box that opens, specify the URL address of the desired Web page. Type the URL address manually or choose a previously specified one from the drop-down list.

Viewing Reference Information

PhpStorm facilitates quick and easy access to the API documentation, which you can view immediately in the editor or in an external browser. To gain access to the API documentation from within your project, make sure to attach archives or directories that contain sources of the library classes.

This section describes how to see definitions of symbols, display documentation references, and use the view parameter information feature:

- Viewing Definition
- Viewing Inline Documentation
- Viewing External Documentation
- Viewing Method Parameter Information

These features are available in all supported language contexts. For information on retrieving reference in JavaScript context, refer to section Viewing JavaScript Reference
.

Viewing Definition

On this page:

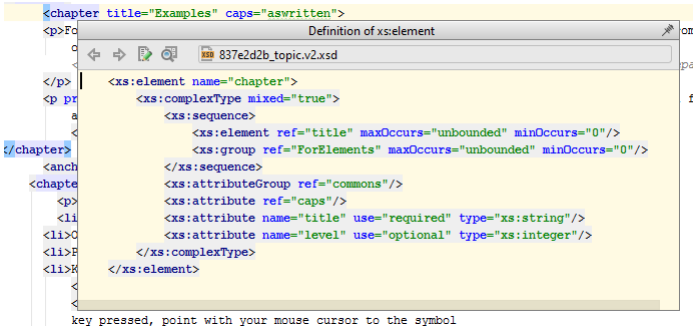
- Basics
- Examples
- Viewing the definition of a symbol at caret
- Toolbar of the quick definition lookup

Basics

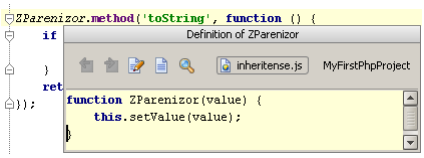
Quick Definition Lookup makes it possible to view definition of a symbol (tag, class, method/function, field, etc.) in a pop-up window.

Examples

For markup languages, PhpStorm retrieves definitions of symbols from the specified DTD or schema. For details, see Markup Languages and StyleSheets .



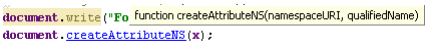
In the JavaScript or PHP context, PhpStorm retrieves the definition of a symbol from the source code and displays the name of the file where the symbol is defined:



Viewing the definition of a symbol at caret

Do one of the following:

- On the main menu, choose View | Quick Definition.
- Press `Ctrl+Shift+I`.
- Keeping the `Ctrl` key pressed, point with your mouse cursor to the symbol of interest, so that it turns to a hyperlink, with the definition of the symbol displayed in a tooltip. Clicking this hyperlink results in opening the respective definition page in the editor.
  Quick definition tooltip shows hyperlinks to the symbols involved.



When you move your mouse pointer within the tooltip, a pin button appears. If you pin the tooltip, documentation for the symbol at caret is displayed in the Documentation Tool Window .

Toolbar of the quick definition lookup

Use the icons on the toolbar of the pop-up window to navigate to the source code of the definition and view its usages.

Icon	Keyboard shortcut	Action
	`Shift+Alt+Left` `Shift+Alt+Right`	Navigate to the previous/next screen in the definition pop-up window after using hyperlinks in the definition. **Note** On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.
	`F4`	Open the source code of the definition for editing, and close the quick definition lookup window. If PhpStorm cannot find the appropriate sources in your project, it will try to decompile the code. In this case the JetBrains Decompiler dialog with legal information is displayed.
	`Ctrl+Enter`	Open the source code of the definition, and preserve the quick definition lookup window opened.

Viewing Inline Documentation

On this page:

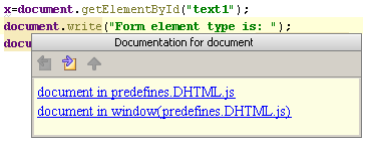
- Basics
- Viewing quick documentation
- Toolbar of the quick documentation lookup

Basics

Quick Documentation Lookup helps you get quick information for any symbol or just method signature information, provided that this symbol has been supplied with documentation comments in the applicable format.

- For markup languages , PhpStorm retrieves reference from the language specification according to the Document Type setting.
- For information on retrieving inline documentation in the JavaScript or PHP context, refer to Viewing JSDoc Comments and Viewing PHPDoc Comments respectively .
- Documentation look-up is also available for Drupal hooks, see Drupal-Specific Coding Assistance .

As a result, such documentation is properly rendered in the Documentation Tool Window :



The URLs and e-mail addresses specified in the documentation comments for methods are also properly rendered. Clicking a hyperlink opens the corresponding URL in an external browser; clicking an e-mail address opens the default mail client.

Viewing quick documentation

To view documentation for a symbol at caret, do one of the following

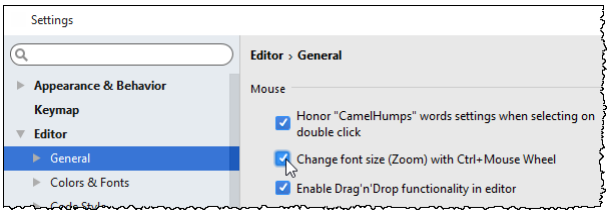
- On the main menu, choose View | Quick Documentation Lookup.
- Press `Ctrl+Q`.
- Provided that the check box Show quick doc on mouse move in the editor settings is selected, just move your mouse pointer over the desired symbol.

When you explicitly invoke code completion, then quick documentation for an entry selected in the suggestion list can be displayed automatically. The behavior of quick documentation lookup is configured in the Code Completion page of the Settings/Preferences dialog.

Tip Sequential pressing `Ctrl+Q` toggles the focus of the Quick Documentation pop-up window and Documentation Tool Window .

To change the font size of quick documentation, do one of the following

- Click in the upper-right corner of the quick documentation window, and move the slider.
- Rotate the mouse wheel while keeping the `Ctrl` key pressed.
  Note that for this feature to work, you have to enable it in the General page of the editor settings.



Refer to the section Zooming in the Editor for details.

Toolbar of the quick documentation lookup

The Quick Documentation Lookup window helps navigate to the related symbols via hyperlinks, and provides a toolbar for moving back and forth through the already navigated pages, changing font size, and viewing documentation in an external browser.

When pinned, the Quick Documentation Lookup turns into Documentation Tool Window , with the corresponding sidebar icon, and more controls.

Icon	Shortcut	Description
	`Left` or `Right`	Switch to the previous or next documentation page (e.g. after using hyperlinks). **Note** On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.

	Shift+F1	View external documentation in the default browser.
	F4	Switch to the item (e.g. source) that corresponds to the documentation page currently shown.
		Turn the Auto-update from source option on or off. When the option is on, the information in the tool window is synchronized with your navigation in the editor and other places in the UI.
		Click this icon to show font size slider. Move the slider to increase or decrease the font size in the quick documentation window as required.

Viewing External Documentation

External documentation makes it possible to get additional information for the symbols at caret. In contrast to the quick documentation , this feature shows the documentation in an external browser, which helps study the symbol in more detail, navigate to related symbols, and retain the information for further reference.

To view documentation for a symbol at caret in an external browser, do one of the following:

- On the main menu, choose View | External documentation.
- Press `Shift+F1`.
- While in the Quick Documentation Lookup window , click .

Viewing Method Parameter Information

This section describes how to configure the View Parameter Information behavior and use this functionality.

To view the method parameters

1. Place the caret anywhere within the call of the desired method or function.

2. Choose View | Parameter Info on the main menu or press `Ctrl+P`.

```
var vehicle = new Vehicle('lamborghini');
                          type
```

Parameter information for methods defined through the `@method` phpDocumentor tag is also available:

```
<?php
/**
 * Test class
 * @method integer method(int $param_1, boolean $param_2 = false) Test method
**/
class Test_Class
{
  public function __call($name, $args)
  {
      echo $name." ".$args;
  }
}
$test = new Test_Class();
param_1 : int , [param_2 : bool = false]
$test->method()
?>
```

To configure the behavior of the view parameter information functionality

1. Open the IDE Settings and click Code Completion below the Editor node.

2. In the Parameter info section, define the following options:
   - To have a complete method or function signature shown rather than a list of required types, select the Show full signatures check box.
     Make sure to include the required third-party JavaScript libraries and PHP-related items in the project source path. Otherwise, names of the parameters will not be displayed.
   - To have the list of parameter types for the called method or function shown automatically after a certain delay, select the Auto pop-up (in ms) check box and specify the time period in milliseconds.

Navigating Through the Source Code

PhpStorm suggests various ways of navigation between the IDE components and within source code. PhpStorm's smart editor makes it possible to navigate across the source code using the code structure rather than plain scrolling. You can find your way in the source code using the method calls, declarations, errors, changes etc.

In this part:

- Navigating with Bookmarks
- Navigating Between Files and Tool Windows
- Navigating Between IDE Components
- Navigating Between Methods and Tags
- Navigating Between Test and Test Subject
- Navigating to Action
- Navigating to Braces
- Navigating to Class, File or Symbol by Name
- Navigating to Custom Folding Regions
- Navigating to Declaration or Type Declaration of a Symbol
- Navigating to Super Method or Implementation
- Navigating to File Path
- Navigating to Line
- Navigating to Next/Previous Change
- Navigating to Next/Previous Error
- Navigating to Recent File
- Navigating to Navigated Items
- Navigating with Navigation Bar
- Navigating with Structure Views
- Navigating with Breadcrumbs

Navigating with Bookmarks

On this page:

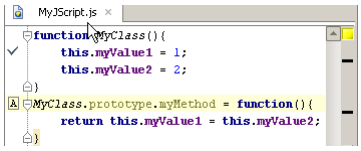
- Introduction
- Navigating within the current file
- Navigating across a project

Introduction

The editor of PhpStorm provides two types of bookmarks:

- Anonymous bookmarks, indicated by check signs ✓ in the left gutter. The number of anonymous bookmarks is unlimited.
- Bookmarks with mnemonics indicated by 1 or S icons in the left gutter. There can be only 10 numbered and 26 lettered bookmarks within a project.

All bookmarks are indicated with the black streaks in the marker bar:



Once created , the bookmarks enable you to easily jump to the desired location within a file, or across the entire project.

Navigating within the current file

To navigate through the bookmarks within the current file, do one of the following

- On the main menu, choose Navigate | Bookmarks | Next/Previous Bookmark. The order of visiting bookmarks depends on their order in the collection of bookmarks in the Bookmarks dialog .
- Click the black streak in the marker bar.

Navigating across a project

To navigate across a project using numbered bookmarks

- Use `Ctrl+Number` where the <number> corresponds to the desired bookmark.

To navigate among all bookmarks in a project, do one of the following

- On the main menu, choose Navigate | Bookmarks | Show Bookmarks, or press `Shift+F11`.
  In the Bookmarks dialog, select the target bookmark, and press `Enter`.

  For your convenience, the target code preview is shown in the right pane of the dialog box.
- In the Favorites tool window , select the desired bookmark in the Bookmarks list, and then double-click the bookmark entry, or press `F4`. The corresponding file opens in the editor, with the caret at the beginning of the bookmarked line.

Managing Bookmarks

This section describes how to:

- Create and delete bookmarks with mnemonics
- Toggle bookmarks
- View project bookmarks
- Delete bookmarks
- Change the order of bookmarks

To create a bookmark with mnemonics

1. Place the caret at the desired line of code in the editor.
2. Press `Ctrl+F11` (alternatively, choose Navigate | Bookmarks | Toggle Bookmark With Mnemonic on the main menu) , then press one of the keys 0-9 or A-Z.

To toggle an anonymous bookmark on the current line

- On the main menu, choose Navigate | Bookmarks | Toggle Bookmark.
- Press `F11`.

To view all bookmarks in a project

- On the main menu, choose Navigate | Bookmarks | Show Bookmarks.
- Press `Shift+F11`.

To delete bookmarks in a project

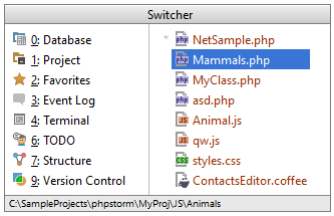
1. Open the Bookmarks dialog (Navigate | Bookmarks | Show Bookmarks, or `Shift+F11`).
2. Select bookmarks and press `Delete`.

To change the order of bookmarks

1. Open the Bookmarks dialog (Navigate | Bookmarks | Show Bookmarks, or `Shift+F11`).
2. Select the desired bookmark.
3. Use Move Up (, `Ctrl+Up`) and Move Down (, `Ctrl+Down`) buttons to shift the bookmark in the desired direction.

Navigating Between Files and Tool Windows

PhpStorm suggests a handy way to switch between files opened in the editor, split editor tabs, and tool windows (docked or floating). This is similar to the application switchers in the various operating systems. The switcher consists of two columns: the left one displays the list of tool windows, while the right one displays the list of files currently opened in the editor.



To switch between open files and tool windows

1. Press `Ctrl+Tab`.

2. Keeping `Ctrl` pressed, use the following keys:
   - `Up` and `Down` arrow keys, `Tab` or `Shift+Tab` to go up and down the list in both panes.
   - `Alt` to toggle between the right and the left panes of the switcher pop-up window.
   - `N/A` or `Backspace` to close the editor tab where the selected file is opened, or hide a tool window, if it is shown.

3. Having selected the desired file or tool window, release the `Ctrl` key. The corresponding file or tool window gets the focus and the switcher pop-up window disappears.

   Tip If you want to close the switcher without selecting any file or tool window, just use the left or right arrow keys.

Navigating Between IDE Components

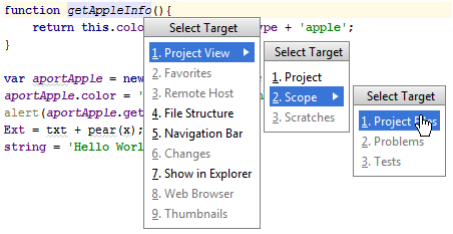
Suppose you have selected a file or member in one of the tool windows, and would like to quickly find it in the Project view. The Select Target pop-up menu moves the focus to the selected component of PhpStorm, the file system, etc.:

- Project View
- Favorites
- Remote Host
- Framework Views
- File Structure
- Database View
- Coverage
- Navigation Bar
- Version Control tool window
- Show in Explorer
- Web Browser
- Thumbnails

To navigate to the desired component

1. On the main menu select Navigate and click Select In, or press `Alt+F1`.
   The Select Target pop-up menu shows up.

2. Use the arrow keys or the mouse pointer to select the desired component. If your target is the Project tool window, you can select the desired view: Project, Scratches etc.



Tip Double-click selected file or member in one of the tool windows, or press `Enter` to open it in the editor.

Navigating Between Methods and Tags

Since your source code can contain numerous methods, it is convenient to navigate to the beginning of the next or previous method. In the Web contents, this feature enables navigating between tags.

To navigate to the next or previous method or tag

- On the main menu, choose Navigate | Next Method / Previous Method respectively.
- Use `Alt+Up` / `Alt+Down` keyboard shortcuts.

For JavaScript code inside HTML files this behavior depends on the caret location. If the caret rests inside a JavaScript block, this means of navigation enables jumping between JavaScript functions. If the caret rests on a `<script>` tag, then navigation is performed between the tags.

To improve the visibility of your source code by having a line added automatically between adjacent methods, choose the Show Method Separators option in the Appearance page of the editor settings.

Navigating Between Test and Test Subject

On this page:

– Overview

– Jumping from a test to its test subject

– Jumping from a class or file to its test

Overview

Testing support in PhpStorm provides the ability to navigate between a test and the test subject. For information on common testing procedures, see Testing . For language-and framework-specific guidelines, see Testing PHP Applications , Unit Testing JavaScript , Running Nodeunit Tests , and Language and Framework - Specific Guidelines .

Jumping from a test to its test subject

1. Open the desired test class in the editor.
2. On the main menu or on the context menu of the editor, choose Navigate | Test Subject. Alternatively, press `Ctrl+Shift+T`.
   The test subject for the current test class opens in the dedicated tab of the editor and gets the focus.

Jumping from a class or file to its test

1. Open the desired class in the editor.
2. On the main menu or on the context menu of the editor, choose Navigate | Test. Alternatively, press `Ctrl+Shift+T`. The test for the current class or file opens in the dedicated tab of the editor and gets the focus.

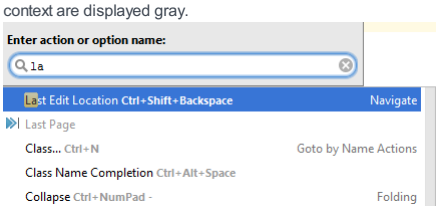
Navigating to Action

PhpStorm helps you quickly find the desired action, without digging through the menus and toolbars. The concept action covers the commands of the main menu and various context menus, commands performed through the toolbar buttons of the main toolbar and tool windows.

To find an action

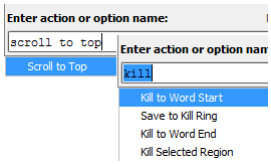
1. Choose Help | Find Action on the main menu or press `Ctrl+Shift+A`. The pop-up window that opens, shows the suggestion list of matching names. By default, this list includes the menu commands only. If you want to include the non-menu commands in the suggestion list, press `Ctrl+Shift+A` once more.

2. Start typing the desired action name. As you type, the suggestion list displays the matching names of actions. The actions that are not valid in the current context are displayed gray.



3. Double-click the desired entry in the suggestion list, or select it using the arrow keys and press `Enter`.

This way you can invoke the actions that are not mapped to keyboard shortcuts in certain schemes (for example, scroll to top, scroll to bottom, or Emacs actions, like kill rings, sticky selection, or hungry backspace).



These actions are not mapped to certain keyboard shortcuts, neither they appear in the menus. If necessary, configure keyboard shortcuts for these actions as described here .

Navigating to Braces

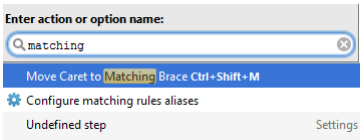
In this section:

- Navigating to the borders of a code block
- Navigating to the borders of the closest higher code block

Navigating to the borders of a code block

To navigate to the borders of a code block, do one of the following:

- To navigate to the code block start, press `Ctrl+Open Bracket`, with the caret anywhere inside the code block.
- To navigate to the code block end, press `Ctrl+Close Bracket`, with the caret anywhere inside the code block.
- To toggle between code block start or end, press `Ctrl+Shift+M`.
  You can also invoke this action (Move Caret to Matching Brace) with a Search Everywhere or Go to Action functionality:



Navigating to the borders of the closest higher code block

To navigate to the borders of the closest higher code block, do one of the following:

- To jump to the higher code block start, press `Ctrl+Open Bracket`, with the caret at the current code block opening brace .
- To jump to the higher code block end, press `Ctrl+Close Bracket`, with the caret at the current code block closing brace .

Tip Practically, you can just press `Ctrl+Open Bracket` or `Ctrl+Close Bracket` as many times as you need, until the caret is positioned at the start or end of the desired code block.

Navigating to Class, File or Symbol by Name

In this section:

- Overview
- Navigating by name
- Tips and tricks

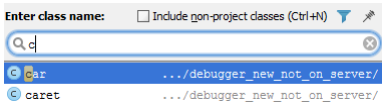
Overview

Navigate commands enable you to quickly jump to the desired classes, files, or symbols specified by names. PhpStorm suggests a look-up list of matching names, from which you can select the desired one, and open it in the editor. This navigation honors CamelCase and snake_case capitalization . Refer to the tips for detailed list of available techniques.

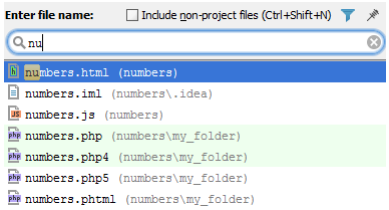
Navigating by name

To navigate to a class, file or symbol with the specified name:

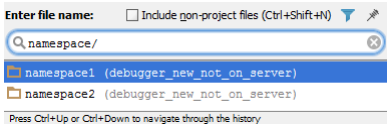
1. On the main menu, point to Navigate, and then choose Class, File, or Symbol respectively, or use the following shortcuts:
   - Class: `Ctrl+N`
   - File (directory): `Ctrl+Shift+N`
   - Symbol: `Ctrl+Shift+Alt+N`
2. In the pop-up window, start typing the desired name.
   So doing, you can enter characters located anywhere inside the desired name. As you type, the suggestion list shrinks, displaying the matching names only.
   - Class:



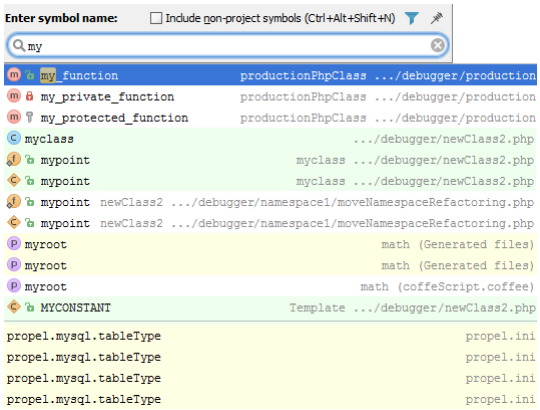
   - File:



   - Directory: use the same `Ctrl+Shift+N` shortcut as for file navigation, and type the name of the directory you are looking for, the pattern name ending with / or \:



   - Symbol:

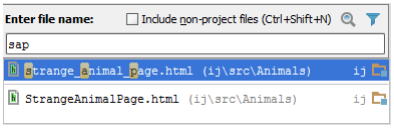


3. Double-click the desired entry in the suggestion list, or select it using the arrow keys, and press `Enter`.

Tips and tricks

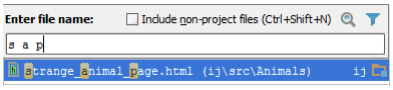
While working in the navigation pop-up window, use the following helpful techniques:

- Narrow down the search scope by selecting the file types to search in. Just click the filter, and clear the check boxes next to the file types you are not interested in.
- Include non-project files in the look-up list and thus make available matching files from SDKs and libraries.
- If the look-up list is too long, type more characters to shrink it, or click the ellipsis sign at the end of the list, to reveal its next portion.
- Type the initial letters of the CamelHumps names, for example:

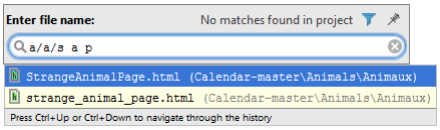


  Note that PhpStorm automatically recognizes CamelHumps and matches them to the lower case letters.

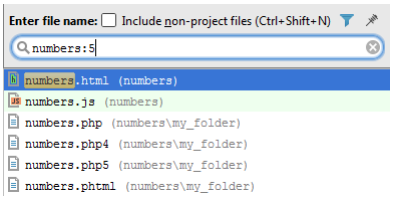
- Type any letters separated with spaces for snake_case names, for example:



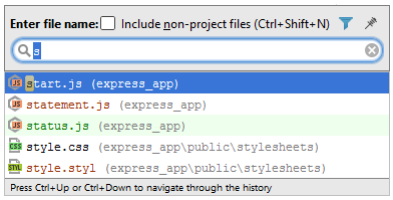
- In the navigation to file pop-up window, type letters delimited with slashes to denote nested directories:



- Type line number after a file name, delimited with a colon, to navigate to the specified line:



- Use * wildcard to represent any number of characters, though it is quite enough to type characters located in the middle of the desired name.
- If while typing in one of the Navigate to Class/File/Symbol pop-up windows you notice that you need another one, just invoke the necessary dialog box. The text you have already entered will not disappear.
- Press `Alt+F1` to invoke the Select Target pop-up window, and choose the desired IDE component.
- Note that for the projects under version control, the entries in the look-up list are color coded according to their status:

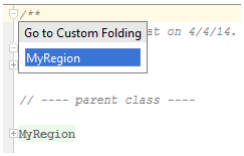


- When there is a detached editor frame with a certain file, you can opt to open this file in the main PhpStorm frame by pressing `Enter`, or activate the detached frame by pressing `Shift+Enter`.

Navigating to Custom Folding Regions

If there are code folding comments in your file (see Using code folding comments), you can navigate to corresponding folding regions like this:

1. Select Navigate | Custom Folding or press `Ctrl+Alt+Period`.
2. Select the target folding region. (The regions in the list are identified by their descriptions.)



Navigating to Declaration or Type Declaration of a Symbol

On this page:

- Introduction
- Important note
- Navigating to the declaration of a symbol
- Navigating to the type declaration of a symbol

Introduction

While editing your source code, you might need to navigate to the location where a particular named code reference (a symbol) has been first declared. Navigate | Declaration command enables you to navigate back to the initial declaration of a symbol from any place in the source code, even if it is from inside another class, or comment.

Important note

Navigate | Declaration/Type Declaration

- Applies to the symbols of source code, CSS, HTML or XML tags and attributes, DTD and schema elements and attributes, and references in comments.
- Does not apply to the primitive types.

Navigating to the declaration of a symbol

To navigate to the declaration of a symbol

1. Place the caret at the desired symbol in the editor.

2. Do one of the following:
   - On the main menu, choose Navigate | Declaration.
   - Press `Ctrl+B`.
   - Click the middle mouse button.
   - Keeping `Ctrl` pressed, point to the symbol, and click, when it turns to a hyperlink. You can also see declaration at the tooltip while keeping `Ctrl` pressed.

```
var vehicle = new Vehicle('lamborghini');
vehicle.c  var vehicle = new Vehicle('lamborghini')
alert(vehicle.getVclInfo());
```

Navigating to the type declaration of a symbol

To navigate to the type declaration of a symbol

1. Place the caret at the desired symbol in the editor.

2. Do one of the following:
   - On the main menu, choose Navigate | Type Declaration.
   - Press `Ctrl+Shift+B`.
   - Press the `Ctrl+Shift` keys and hover your mouse pointer over the symbol. When the symbol turns to a hyperlink, click it without releasing `Ctrl+Shift` keys. The type declaration of the symbol opens in the editor. You can also see the declaration at the tooltip while keeping `Ctrl+Shift` pressed.

```
                    function Vehicle(type)
var vehicle = new Vehicle('lamborghini');
```

Navigating to Super Method or Implementation

PhpStorm provides an easy way to navigate up and down through the hierarchy of methods. If a method is overridden by a certain method, or overrides some method itself, it is marked with an icon in the gutter area of the editor. When the mouse cursor hovers over such icon, the method information is displayed as the tooltip:

> **Tip** In the PHP context, PhpStorm also indicates class fields that override the corresponding fields of the superclass and the fields that are overridden in the subclasses.

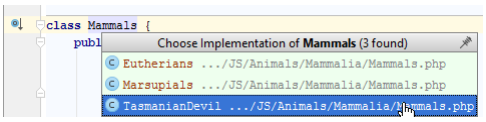
- : This method or field overrides a method defined by its superclass.
- : This method or field is overridden by one or more subclasses.

Use these icons, shortcuts, or menu commands to navigate to the corresponding points of origin.

Navigating through the hierarchy of methods

To navigate up and down through the method hierarchy, do one of the following:

- Click the gutter icon and select the desired ascendant or descendant class from the list.
- On the main Navigate menu, choose Super Method, or Implementation(s) respectively.
- Press `Ctrl+U` or `Ctrl+Alt+B` for the super method or implementation respectively.



Navigating to File Path

On this page:

– Overview

– Navigating to a file path

– Viewing file path

Overview

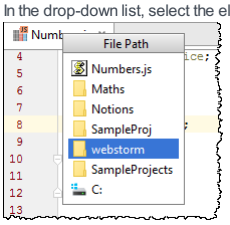
It may be helpful to open a file in a file manager (for example, in the Windows Explorer, if Windows is your OS), or the Finder. PhpStorm allows you to easily navigate to any part of a file path, right from the editor.

Navigating to a file path

To navigate to a file path from the editor

1. Hold Ctrl / ⌘ and click the relevant tab in the editor.

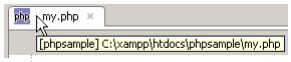
2. In the drop-down list, select the element you need.



The file manager shows the selected file or directory.

Viewing file path

To just view the path to a file, place the mouse pointer over the tab where the file is opened.



Navigating to Line

Navigate | Line command is the most basic way to navigate to the line with the specified number.

To navigate to a line in the editor

1. On the main menu, choose Navigate | Line, or press `Ctrl+G`.
2. In the Go To Line dialog box that appears, PhpStorm shows the current line and offset delimited by a colon. Type the target line number and, optionally, an offset, and then click OK.

The current caret position (line number and offset) is displayed in the Status Bar . You can click it to open the Go To Line dialog box.

Navigating to Next/Previous Change

If you edit a file that is under version control, PhpStorm provides several ways to move back and forth with the updates. In particular, you can use the navigation commands, keyboard shortcuts, and the change markers.

To navigate to the next/previous change in the editor, do one of the following:

- On the main menu, choose Navigate | Next / Previous Change.
- Use keyboard shortcuts `Ctrl+Shift+Alt+Down` or `Ctrl+Shift+Alt+Up`.
- Point to a change marker , and click the arrow up or arrow down buttons.

To navigate to the place of your last edit, do one of the following:

- On the main menu, choose Navigate | Last Edit Location
- Use keyboard shortcut `Ctrl+Shift+Backspace`.

Navigating to Next/Previous Error

Another method of code navigation is to move between found errors and warnings. The caret is positioned immediately before the code issue.

You can configure the way PhpStorm navigates between code issues: it can either jump between all code issues or skip minor issues and only navigate between detected errors.

In this section:

- Configuring the error navigation
- Navigating between errors or warnings

To configure the error navigation

1. Right click the Validation Side Bar .
2. On the context menu, choose one of the available navigation modes:
   - To have PhpStorm skip warnings, infos, and other minor issues, choose Go to high priority problems only.
   - To have PhpStorm jump between all detected code issues, choose Go to next problem.

To navigate between errors or warnings, do one of the following

- On the main menu, choose Navigate | Next / Previous Highlighted Error.
- Use keyboard shortcuts `F2` and `Shift+F2` respectively.

Navigating to Recent File

In this section:

- Navigating to a recently opened file
- Navigating to a recently edited file
- Navigating to the latest edit location
- Navigating to the next edit location
- Using multi-selection in the lists of recent files

To navigate to a recently opened file

1. On the main menu, choose View | Recent Files or press `Ctrl+E`.
2. From the Recent Files pop-up window that opens select the desired file.

To navigate to a recently edited file

1. On the main menu, choose View | Recently Changed Files or press `Ctrl+Shift+E`.
2. From the Recently Edited Files pop-up window that opens select the desired file.

Tip The recently opened or recently modified files are selected from the history list. The number of entries in the history list is configurable in the Recent file limit field in the Editor settings page.
- Navigating to recent files applies to the search results as well. By pressing `Ctrl+E` in the Find tool window, you can have the list of recent search results shown.

To jump to the latest edit location

- Do one of the following:
  - On the main menu, choose Navigate | Last Edit Location.
  - Press `Ctrl+Shift+Backspace`.

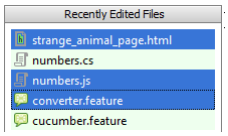
To jump to the next edit location

- On the main menu, choose Navigate | Next Edit Location.

This action is not bound to any shortcut. An interested user can do it as described in the section Configuring Keyboard Shortcuts .

Using multi-selection in the lists of recent files

- To select non-adjacent files, use `Ctrl` + mouse click.
- To select adjacent files, use `Shift` + mouse click.



Navigating to Navigated Items

You can go back and forth along the items that have already been navigated to. This feature applies to the items reached using all navigation commands except for the simplest ones, such as arrow keys, Pageup, Page Down, Home and End.

To navigate to the navigated items

Do one of the following:

- On the main menu, choose Navigate | Back / Forward.
- Use keyboard shortcuts `Ctrl+Alt+Left`, or `Ctrl+Alt+Right`.
- On the main toolbar, click or .

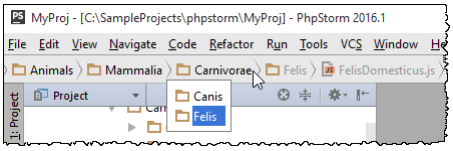
Note On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.

Navigating with Navigation Bar

Use the Navigation Bar as a handy tool to find your way across the project.

To navigate to a file using the Navigation bar

1. Press `Alt+Home` to activate the Navigation bar.
2. Use the arrow keys or the mouse pointer to locate the desired file.
3. Double-click the selected file, or press `Enter` to open it in the editor.



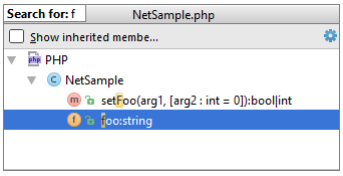
Navigating with Structure Views

Use the Structure pop-up window or the Structure tool window to quickly jump to the desired member of a file in the editor. The Structure views provide quick navigation for all supported file types.

To navigate to a member in the editor

1. Choose Navigate | File Structure on the main menu or press `Ctrl+F12`.

2. In the File Structure pop-up window that opens select the Narrow down the list on typing check box and start typing the desired member name.
   You can include the members, inherited from the parent classes, by checking the option Show inherited members or pressing `Ctrl+F12` again.



3. Use the navigation keys to select the desired node. Then do one of the following:
   - If the cursor rests on a top or intermediate node (for example, class or element), double-click this node or press `Enter` to expand it in the Structure pop-up, or press `F4` to jump to its declaration in the editor.
   - If the cursor rests on a leaf node (for example, a member , or lowest-level element), double-click this node or press `Enter` to jump to its declaration in the editor.

   In the case of an inherited member, the respective parent class opens in the editor.

You can also use the Structure tool window (`Alt+7`). This view is flexibly configurable and useful for many tasks, apart from navigation. However, the File Structure pop-up window is the easiest way for quick navigation.

Navigating with Breadcrumbs

On this page:

- Overview
- Enabling breadcrumbs
- Navigating between a breadcrumb and the source code

Overview

In the HTML and XML context and in PHP classes, you can navigate through the source code using breadcrumbs that are displayed above the main editor window. The breadcrumbs show the elements of a XML/HTML file or PHP class opened in the active editor tab :

To have breadcrumbs shown, you have to enable them.

Enabling breadcrumbs

To have breadcrumbs displayed

1. Open the PhpStorm Settings/Preferences dialog.
2. Under the Editor node, click General | Appearance.
3. In the Appearance page, select the Show breadcrumbs check box.

Navigating between a breadcrumb and the source code

To navigate with breadcrumbs

- Click the breadcrumb with the relevant tag or the PHP class element. The cursor jumps to the corresponding element in the source code.

Searching Through the Source Code

PhpStorm provides extensive search and replace capabilities, which includes basic search and replace, search and replace in paths, finding usages, and more. Some of the replacement facilities (like renaming classes and members) are performed by means of refactorings .

All the search commands can be found under the Find node of the Edit menu.

In this part you will learn how to:

- Find and replace text in the current file.
- Find specific word .
- Search and replace across the project .
- Find usages .
- Highlight usages .
- View usages of a symbol across the project, and jump to the desired usage .
- Work with the search results .

Finding and Replacing Text in File

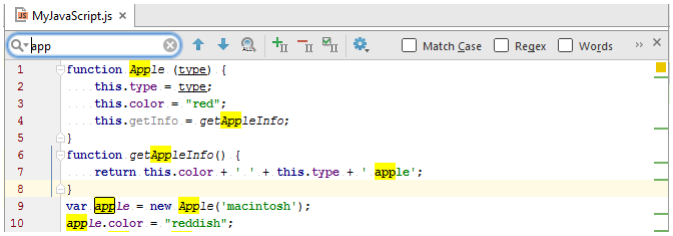
The standard facility helps you find and replace text strings in the active editor.

On this page:

- Search through the current file
- Replace in the current file
- Working with search results
- Search and replace options

Search through the current file

1. From the main menu, choose Edit | Find | Find, or press `Ctrl+F`. The search pane appears on top of the active editor.
2. If necessary, specify the search options .
3. In the search field, start typing the search string:



As you type, the first occurrence of the search string after the current cursor position is selected; the other occurrences are highlighted in the editor. In addition, the matching occurrences are marked in the right gutter with stripes.

4. Explore the search results .

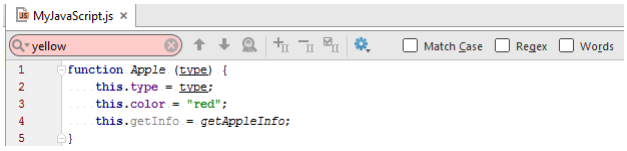
Replace in the current file

1. From the main menu, choose Edit | Find | Replace, or press `Ctrl+R`. The search and replace pane appears on top of the active editor.
2. If necessary, specify the search and replace options .
3. In the search field, start typing the search string. As you type, the matching occurrences are highlighted in the editor, and a Replace pop-up dialog box opens at the first occurrence, suggesting you to replace the current occurrence, or all of them, with an empty string.
4. Start typing the replacing string.
5. Explore the search results , and, using the buttons of the replace dialog box, replace occurrences as required. See Search and replace options below.

Working with search results

- To initiate a new search, do one of the following (depending on the current focus):
  - If the editor has the focus, press `Ctrl+F`.
  - If the search field has the focus, press `Ctrl+A`

  In both cases, the existing search string will be selected, and you can start typing a new one.
- To jump between occurrences, do one of the following:
  - Press `Shift+F3` (jump to previous selection) or `F3` (jump to next selection).
  - Use the ↑ or ↓ buttons in the Search pane.
  - Click the gutter stripes.
- The search pane shows the number of found occurrences. If no matches are found, the search pane is highlighted with red:



- Use Code completion in the Find and Replace panes. Start typing a search string, press `Ctrl+Space`, and select the appropriate word from the suggestion list.
- Use the recent search history: with the search pane already open, click to show the list of recent entries.
- With the Find and Replace pane already opened, use `Ctrl+R` or `Ctrl+F` to toggle between panes. Thus, the search and replace strings are preserved.
- To cancel the operation and close the pane, press `Escape`.
- Use multiple selection (multiselection). For example, if a certain string has been highlighted as a search result, it is possible to add an occurrence of this string to multiple selection by clicking (`Alt+J`), delete an occurrence from multiple selection using (`Shift+Alt+J`), or add all found occurrences to multiple selection using (`Ctrl+Shift+Alt+J`).

Search and replace options

Item	Description	Search/Replace
	Click this button to show the history of the recent entries.	Search, replace
	Click this button to clear the search field.	Search, replace
↑ ↓	Click these buttons to navigate through the occurrences of the search string.	Search, replace
	Click this button to add the next found occurrence to a multiple selection.	Search
	Click this button to remove the found occurrence from a multiple selection.	Search

	Click this button to create a selection that contains all the found occurrences.	Search
	Click this button to show search results in the Find tool window .	Search, replace
Match Case	If this check box is selected, PhpStorm will distinguish between upper and lowercase letters while searching.	Search, replace
Regex	If this check box is selected, the search string will be perceived as a regular expression , and the replacement preview is shown in a tooltip (Refer to the section Example).	Search, replace
Words	if this check box is selected, PhpStorm will search for whole words only, that is, for character strings separated with spaces, tabs, punctuation, or special characters. This check box is disabled, if the Regular expressions check box is selected.	Search, replace
Preserve Case	If this check box is selected, PhpStorm retains the case of the first letter and the case of the initial string in general. For example, *MyTest* will be replaced with *Yourtest* if you specify *yourtest* as the replacement. This check box is disabled, if Regular expressions check box is selected.	Replace
In Selection	If this check box is selected, search and replacement will be confined to the selected text only.	Replace
Replace	Click this button to replace the current occurrence and proceed to the next one.	Replace
Replace all	Click this button the replace all found occurrences in the current file, or in the selection.	Replace
Exclude/Include	Click Exclude button to skip the current occurrence and exclude it from the Replace all operation. The button for this occurrence changes to Include.	Replace
	Click this button to invoke the list of additional options. Checking the corresponding option confines the search to the specified scope, while the other occurrences are ignored.	Search, replace

Finding and Replacing Text in Project

PhpStorm extends search and replace capability to the entire project, or any directory with its nested hierarchy. Explore search results in the Find tool window .

On this page:

- Finding a piece of text in all the files within the specified path
- Replacing a piece of text in all the files within the specified path
- Toggling between the Find in Path and Replace in Path dialog boxes

Finding a piece of text in all the files within the specified path

1. On the main menu, choose Edit | Find | Find in Path, or press `Ctrl+Shift+F` .

2. In the Find In Path dialog box, specify the following options:
   - The text to find. Type the text explicitly, or specify a pattern using a regular expression, or select a previously used piece of text or a pattern from the recent history drop-down list.
     If you specify the search pattern through a regular expression, use the `$n` format in back references (to refer to a previously found and saved pattern).
   - Search scope (project , module or directory).
   - Search options (case sensitivity, whole words, and regular expressions).

3. Click Find.

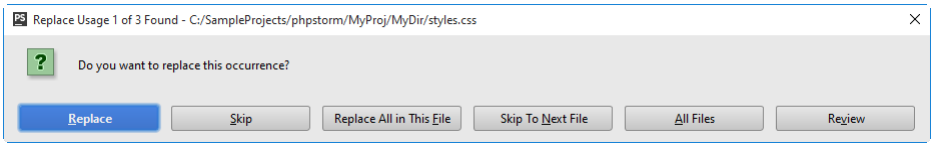
If the search takes too long time, click Background in the search progress window. In this case the search progress is indicated in the Status bar.

Replacing a piece of text in all the files within the specified path

1. Do one of the following:
   - On the main menu, choose Edit | Find | Replace in Path.
   - Press `Ctrl+Shift+R`
   - Being in the Find In Path dialog box, press `Ctrl+Shift+R` to switch to Replace In Path dialog box.

2. In the Replace In Path dialog box, specify the search and replace strings, the search options, and the scope. Type the search and replacement text explicitly, or specify patterns using regular expression, or select a previously used piece of text or a pattern from the recent history drop-down list.
   - If you specify the search and/or replacement text through a regular expression, use the `$n` format in back references (to refer to a previously found and saved pattern).
   - To use a backslash character `\` in a regular expression, escape the meaningful backslash by inserting three extra backslashes in preposition: `\\\\` .

3. Click Find. PhpStorm displays the encountered occurrences of the search string in the Find tool window , selects the first occurrence and opens the file with this occurrence in the editor and moves the focus to it.
   At the same time, PhpStorm opens the Replace Usage dialog box, with the full path to the encountered occurrence in the title bar:



   Do one of the following:
   - To have the selected occurrence replaced, click Replace.
   - To preserve the selected occurrence and move to the next one, click Skip.
   - To have all the occurrences of the search string in the currently active tab replaced, click Replace All in This File.
   - To preserve the occurrences of the search string in the currently active tab (any) and move to the next file, click Skip to Next File.
   - To have all the detected occurrences replaced, click All Files.
   - To switch to the manual mode, click Preview. The Replace Usage dialog box closes and the focus moves to the Find tool window. Do one of the following:
     - Browse through the list of detected occurrences, select the ones you want to replace and then click Replace Selected.
     - To have all the occurrences changed click Replace All.

Tip To learn about changing case of the literals, refer to the example , and to Regular Expression Syntax Reference .

Toggling between the Find in Path and Replace in Path dialog boxes

- To switch from the Find In Path to Replace In Path dialog box, press `Ctrl+Shift+R` . To switch from the Replace In Path to Find In Path dialog box, press `Ctrl+Shift+F` .

Finding Usages

Search for usages is an important part of the code analysis, which enables you to clarify dependencies. PhpStorm suggests several types of search for usages:

- Finding Usages in Project
- Finding Usages in the Current File
- Highlighting Usages
- Viewing Usages of a Symbol
- Viewing Recent Find Usages

Finding Usages in Project

PhpStorm provides different search options depending on whether you are searching for usages of a class, method, field, parameter, or throw statements, and extends search for usages to the files in supported languages. For example, in CSS, XML and HTML files you can search for the usages of styles, classes, tags and attributes.

Explore search results in the Find tool window .

Finding usages of a symbol in a project

1. Select a symbol to find usages for. To do that, place the caret within the desired symbol in the editor, or click the symbol in the Project tool window . You can also select symbol in the
2. Do one of the following:
   - On the main menu, choose Edit | Find | Find Usages
   - Choose Find Usages on the context menu
   - Press `Alt+F7`.
3. In the Find tool window , explore search results. Use the button to represent search results in meaningful groups by type of usage.

While analyzing the search results, you can at any time open the search options dialog box by clicking click in the Find tool window or by pressing `Ctrl+Shift+Alt+F7`.

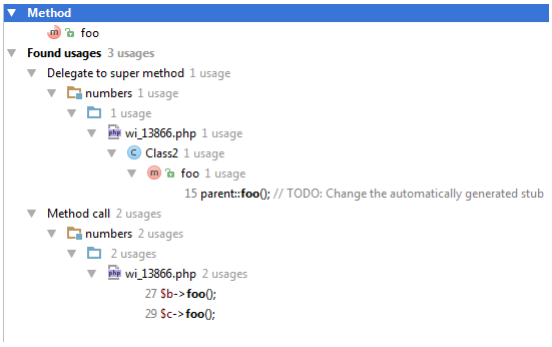
Finding usages of implemented and overridden methods

In the PHP context, PhpStorm also applies the Find Usages functionality to implemented and overridden methods. Consider the following example:

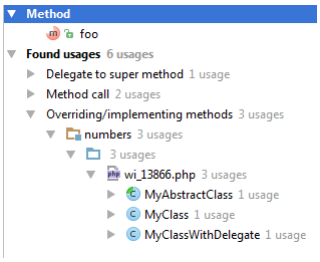
1. Create an interface, an abstract class that implements it, and two classes that extend the abstract class:
   1. Create an interface `MyInterface` with a `foo()` method.
   2. Create an abstract class `MyAbstractClass` that implements `MyInterface`.
   3. Create a class `MyClass` that extends `MyAbstractClass` and implement the `foo()` method required by the interface and overrides the methods of the parent class.
   4. Create a class `MyClassWithDelegate` that extends `MyClass` and implement `foo()` with a delegate.
   5. Create variables `$b` and `$c` that call `foo()` from `MyClass` and `MyClassWithDelegate` respectively:

```php
<?php
interface MyInterface {
//press Alt-F7 on foo() here
public function foo();
}

abstract class MyAbstractClass implements
MyInterface {
public function foo () {
// TODO: Implement foo() method.
}
}

class MyClass extends MyAbstractClass
{
public function foo() {
parent::foo(); //
TODO: Change the automatically generated stub
echo "foo";
}
}

class MyClassWithDelegate extends MyClass
{
public function foo() {
foo();
}
}

$b = new MyClass();
$b->foo();
$c = new MyClassWithDelegate();
$c->foo();
```

2. From `MyInterface`, invoke Find Usages for `foo()` by pressing `Alt+F7` or choosing Edit | Find | Find Usages on the main menu. By default, PhpStorm shows only delegates to the super method and method calls:



3. To find also the methods that implement or override the base method, click in the Find tool window . Then in the Find Usages. Method Options dialog that opens, select the Include overloaded methods check box and click Find. As a result, all the usages of the `foo()` method are found in all the classes that implement or extend `MyInterface` :



To find usages of a symbol in the current file

1. Click the desired symbol in the editor, or in the Structure view.
2. On the main menu, choose Edit | Find | Find Usages in File, or press `Ctrl+F7`. The encountered usage is highlighted in the editor.

Highlighting Usages

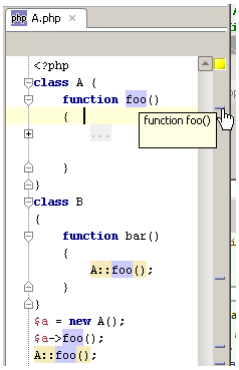
On this page:

- Introduction
- Configuring automatic highlighting
- Highlighting usages in the current file
- Navigating among usages
- Removing highlighting

Introduction

The search command Highlight Usages in File (`Ctrl+Shift+F7`) makes it possible to visualize usages of a symbol in the current file.

All found usages of a symbol in the current file are highlighted and color-coded, as defined in the Colors and Fonts settings page, to represent read or write access to the symbol. In addition to the highlights of occurrences in text, the stripes of the same colors appear in the marker bar, accompanied with tooltips.



The behavior of usage highlighting is configurable: you can make PhpStorm show usages of a symbol at caret automatically, or invoke it with a command.

Configuring automatic highlighting

To configure highlight usages behavior

1. Open the Settings/Preferences dialog .
2. In the Editor page of the IDE Settings, select the check box Highlight usages of element at caret to enable automatic usage highlighting.

Highlighting usages in the current file

To highlight usages of a symbol in the current file

1. Place the caret at the selected symbol in the editor. If automatic usages highlighting is enabled, see all its occurrences in the current file highlighted. Otherwise, proceed to the next step.
2. On the main menu, choose Edit | Find | Highlight Usages in File, or press `Ctrl+Shift+F7` .

Navigating among usages

To navigate among usages, do one of the following

- Click on a stripe in the marker bar to navigate to the respective usage location.
- Use `F3` and `Shift+F3` keyboard shortcuts, to navigate to the next and previous usages respectively.

Removing highlighting

To remove highlighting of usages

- Press `Escape` .

Viewing Usages of a Symbol

Using the Show Usages function, you can bring up a list of the usages of a symbol across the whole project. So doing, the pop-up window with the list of usages of a symbol features a toolbar with the following buttons:

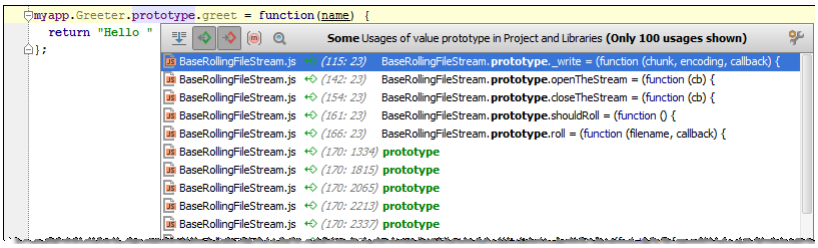
Icon	Shortcut	Description
	Ctrl+F	Merge usages of the symbol from the same line.
	Ctrl+R	Show read access to the symbol.
	Ctrl+W	Show write access to the symbol.
	Ctrl+I	Show usages in the import statements.
	Ctrl+M	If this button is pressed, the found usages show under the corresponding method nodes.
	Alt+F7	Show search results in the Find Usages tool window.
	Ctrl+Shift+Alt+F7	Open the Find Usages dialog box for the selected symbol where you can change the search options.

Tip In addition to the ability of viewing usages, you can use this function as a quick means of navigation.

To view the usages of a symbol across the project

1. Place the caret at the desired symbol in the editor.

2. On the main menu, choose Edit | Find | Show Usages, or press Ctrl+Alt+F7. PhpStorm displays the Show Usages pop-up window which lists all occurrences of the symbol at caret.



3. Examine and analyze the detected occurrences of the symbol:
   - Use the toolbar buttons to present search results in the desired way.
   - To jump from search results to a line of source code, click the desired entry.
   - To close the list, press Escape.

4. If necessary, customize the search options in the search options dialog box. To invoke the dialog box, do one of the following:
   - In the Show Usages pop-up window, click .
   - Press Ctrl+Shift+Alt+F7.

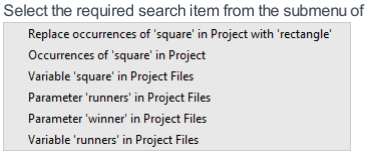
Viewing Recent Find Usages

With PhpStorm you can easily navigate to the recent search results if any find usages action took place during the PhpStorm session.

To view recent find usages

1. Select Edit | Find | Recent Find Usages on the main menu.

2. Select the required search item from the submenu of the recent search results:

   The selected search shows in the Find tool window.

Finding Word at Caret

Search for a word at caret enables you to quickly find the exact match for the current word, without changing any search options. With the match once found, you can navigate between the occurrences of the term.

To find the word at caret, do one of the following

- On the main menu, choose Edit | Find | Find Word At Caret.
- Use `Ctrl+F3` keyboard shortcut.

To navigate between the occurrences of the word at caret

1. Press `F3` to go to the next occurrence.
2. Press `Shift+F3` to go to the previous occurrence.

Working With Search Results

The search results display in the Find tool window . The results of each search display in a separate tab.

Using the controls and the context menu commands of this tool window as well as the main menu, you can:

- Navigate to source code .
- Exclude and include search results in refactoring.
- Add selected search results to favorites .
- Show the results of the recent find usages.
- Perform version control operations using the specific VCS, associated with the parent directory of the usage.
- Hide excluded search results from showing them in the Find tool window (`Alt+Delete`) .

Searching Everywhere

On this page:

- Introduction
- Searching everywhere
- Configuring search everywhere scopes

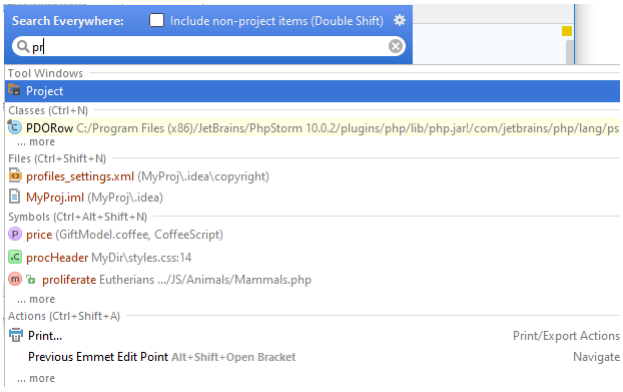
Introduction

PhpStorm makes it possible to look for any item of the source code, databases, actions, elements of the user interface, etc. in a single action.

Searching everywhere

To search everywhere, follow these steps

1. Do one of the following:
   - Click in the upper-right corner of the PhpStorm window.
   - Double-press `Shift`
2. In the pop-up window that opens, start typing the search string, to narrow down the suggestion list:



   The search string can be a name of a symbol or action, an option, or an abbreviation of an action.

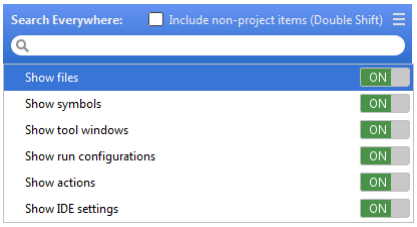
3. Select the desired target from the suggestion list. The further behavior depends on the search target. PhpStorm may silently navigate to the desired source code, show the desired setting, or a pop-up list of Find Action.
   You can also just press the desired shortcut, to switch from the Search everywhere pop-up window to any other navigation and search action: for example, press `Ctrl+N` to jump to the selected class, or `Ctrl+Alt+S` to show the corresponding option in the Settings dialog.

Tip PhpStorm allows viewing the history of the previous searches with the help of the left arrow key. Search Everywhere remembers last 50 queries finalized with `Enter` key.

Configuring search everywhere scopes

To configure search everywhere scopes

1. In the Search Everywhere dialog, click .
2. Turn on or off the desired search scopes:



Structural Search and Replace

Structural Search and Replace (SSR) performs search and replace in the supported languages code across the whole project, taking advantage of the PhpStorm's awareness of the syntax and code structure of the supported languages.

PhpStorm finds, and if required, replaces fragments of source code, using the search templates .

Structural Search and Replace is helpful when you need to browse through or modify an extensive code base, find changes in libraries, explore the source code for specific constructs, or refactor the source code.

> **Tip** To see the list of supported languages, open the dialog box Structural Search and Replace Dialogs and click the drop-down list File type.

In this section:

- Search Templates
- Structural Search and Replace - General Procedure
- Creating and Editing Search Templates
- Structural Search and Replace Examples

Search Templates

Search templates are the essential part of Structural Search and Replace feature. Like Live templates , the search templates consist of plain text and one or more template variables.

A valid search or replacement template represents one of the following supported languages constructs:

- Expression, for example `"John" + " " + "Doe"`
- Statement, or sequence of statements, for example `document.getElementById("demo").innerHTML = "Hello Dolly.";`
- Class designator, for example `class Engine implements IEngine`
- Line or block comments, for example `/** Created in PhpStorm */` .

PhpStorm provides a collection of predefined search templates, that match the various statements, expressions, classes and their members, XML and HTML constructs, and more. You can use these templates for structural search and replace, and also as a basis for creating your own search templates.

The search templates make use of the variables, which are the strings surrounded with `$` characters, for example `$expression$` . Symbols in the source code, `String` literals, and comments can be referred to by means of variables.

Variables in a template are subjected to certain constraints that help you refine your search and limit it to the desired matches:

- Text constraints are text patterns to match against. These constraints can be plain text, or regular expressions, and can contain references to symbols.
- Number of occurrences defines how many sequential elements (in a parameter, declaration or statement list) a variable can include and whether a variable is required to be present in a pattern or not. If the number of occurrences is 1, only one symbol can match the variable. If the number of occurrences is null, it means that an element could be missing.
- Expression constraints apply semantic conditions to the search, for example locate the symbols that are read or written to.
- Script constraints are used when items to search for are more than a plain match. If you are looking for certain language constructs (for example, constructors with the specified number of parameters, or members with the specified visibility modifiers), apply constraints described as Groovy scripts.

In search templates, the following simplifications can be used:

- Method body can be omitted.
- Short class names (instead of fully qualified names) are used in the templates and constraint fields.
- Using `class $Class$` as a template, results in finding anonymous classes as well.
- Templates for comments and documentation comments should contain variables and constructs with correct comment and PhpDoc/JSDoc syntax.

Structural Search and Replace - General Procedure

This section outlines the general SSR procedure. Refer to the section Structural Search and Replace Examples for typical use cases.

To find and replace source code structurally, follow these general steps:

1. On the main menu, choose Edit | Find | Search Structurally, or Edit | Find | Replace Structurally.

2. In the dialog box that opens, define the search template . In brief, defining a search template involves the following steps:
   - Type the desired construct in the Search template text area, or use one of the pre-defined search templates by clicking the Copy existing template button.
   - Specify the constraints to be imposed on the variables within the search template. To do that, click the Edit variables button. All the variables contained in the search template are listed in the Variables pane of the Edit Variable dialog box.

   Refer to the section Creating and Editing Search Templates for the detailed description of procedure.

3. In case of the structural replace, specify the replacement pattern, and define variable constraints as required.

4. Specify the search and replace options, in particular, the number of occurrences to be matched, and the type of files to be analyzed.

5. Specify the scope to perform the structural search and replace in. To do that, click the down arrow in the Scope list, and select one of the pre-defined scopes, or click the ellipsis button, and configure the desired scope in the Scopes dialog box.

6. Click Find. The detected occurrences are displayed in the Find tool window .
   Please note that in case of replacement, you can select the desired matches in the search results and click the Preview Replacement button. The corresponding occurrence is highlighted in the source code.

Creating and Editing Search Templates

You can create search and replace templates from scratch, just typing the code in the text area of the Structural Search / Replace dialog box. However, there is a collection of the predefined search templates that you can use as prototypes for your own templates. All custom templates appear in the list of existing search templates, under the node User defined.

To create a search template, follow these general steps:

- On the main menu, choose Edit | Find | Search Structurally.
- Do one of the following:
  - Type the code of your template in the Search template text area.
  - Click the Copy existing template button, and in the Existing Templates dialog box, select the desired template as a prototype. The source code of the selected template appears in the Search template text area, where you can change it as required.
- If you need to configure the template variables, click the Edit variables button. Edit Variables dialog box appears.
  In the Variables column, select a variable you want to configure, and specify the constraints that will apply to this variable. See the dialog reference page for the detailed description of constraints.

  Repeat the process for the other variables, as required, apply changes and close the dialog box.
- Click the Save Template button.
- In the Save Template dialog, type the name of the new template, and click OK.

Tip You can create a new template in the Existing Templates dialog. To do that, click the + button on the toolbar. This opens the Structural Search dialog, with the empty template field. To define the custom template, follow the steps described above.

Searching for PHP classes

If you have, for example, a PHP class:

```
<?php
class MyClass {
}
```

the simplest template to search for a class is:

```
<$a$/>
```

Searching for XML and HTML tags, attributes, and their values

The simplest template that is used to search for a tag is:

```
<$a$/>
```

By placing constraints on the variable `$a$`, you can specify which tags you want to find. For example, if you specify the text/regexp constraint `app.+`, you'll find the tags whose names start with `app`.

A more versatile template for searching in XML and HTML is:

```
<$tag$ $attribute$="$value$"/>
```

By using this template with properly specified search settings and constraints, you can find practically anything that may occur in XML or HTML. For example, if you specify the text/regexp constraint `width` for the variable `$attribute$`, you'll find all the tags that have the `width` attribute.

Refactoring Source Code

PhpStorm offers a wide variety of code refactorings, which track down and correct the affected code references automatically.

To perform refactoring, follow these general steps

1. Select (or hover caret on) a symbol or code fragment to refactor. The set of available refactorings depends on your selection. You can select symbols in the following PhpStorm components:
   - Project view
   - Structure tool window
   - Editor
   - UML Class diagram

2. Do one of the following:
   - On the main Refactor menu or on the context menu of the selection, choose the desired refactoring or press the corresponding keyboard shortcut (if any).
   - On the main menu, choose Refactor | Refactor This, or press `Ctrl+Shift+Alt+T`, and then select the desired refactoring from the pop-up window.

3. In the dialog box that opens, specify the refactoring options.

4. To apply the changes immediately, depending on the refactoring type, click Refactor or OK.

5. For certain refactorings, there is an option of previewing the changes prior to actually performing the refactoring. In such cases the Preview button is available in the corresponding dialog.
   To preview the potential changes and make the necessary adjustments, click Preview. PhpStorm displays the changes that are going to be made on a dedicated tab of the Find tool window .

   One of the possible actions at this step is to exclude certain entries from the refactoring. To do so, select the desired entry in the list and press `Delete`.

   If conflicts are expected after the refactoring, PhpStorm displays a dialog with a brief description of the encountered problems. If this is the case, do one of the following:
   - Ignore the conflicts by clicking the Continue button. As a result, the refactoring will be performed, however, this may lead to erroneous results.
   - Preview the conflicts by clicking the Show in View button. PhpStorm shows all conflicting entries on the Conflicts tab in the Find tool window , enabling you to navigate to the problematic lines of code and to make the necessary fixes.
   - Cancel the refactoring and return to the editor.

6. When you are satisfied with the proposed results, click Do Refactor to apply the changes.

PhpStorm provides the following common refactorings:

- Change Signature
- Copy/Clone
- Extract Constant
- Extract Field
- Extract Interface
- Extract Method
- Extract Parameter
- Extract Variable
- Extract Variable for Sass
- Inline
- Make Method Static
- Move Refactorings
- Pull Members up
- Push Members down
- Rename Refactorings
- Safe Delete

The following language-specific refactorings are available:

- Extract Parameter in JavaScript
- Change Signature in JavaScript
- Extract Variable in JavaScript

Change Signature

In this section:

- Basics
- Default value and propagation of new parameters
- Examples
- Changing a function signature

Basics

The Change Signature refactoring combines several different modifications that can be applied to a function signature. You can use this refactoring for the following purposes:

- To change the function name.
- To add new parameters and remove the existing ones.
- To assign default values to the parameters.
- To reorder parameters.
- To change parameter names.
- To propagate new parameters through the function call hierarchy.

When changing a function signature, PhpStorm searches for all usages of the function and updates all the calls, implementations, and override replacements of the function that can be safely modified to reflect the change.

> **Note** The Change Method Signature refactoring is supported only for PHP and JavaScript (see Change Signature in JavaScript .
> - You can access this refactoring from a UML Class diagram .

Default value and propagation of new parameters

For each new parameter added to a function, you can specify:

- A default value (or an expression) (the Default value field).

You can also propagate the parameters you have introduced to the functions that call the function whose signature you are changing.

The refactoring result depends on whether you specify the default value and whether you use propagation.

Propagation. New parameters can be propagated to any function that calls the function whose signature you are changing. In such case, generally, the signatures of the calling functions change accordingly.

These changes, however, also depend on the default values set for the new parameters .

Default value. Generally, this is the value to be added to the function calls.

If the new parameter is not propagated to a calling function, the calls within such function will also use this value.

If the propagation is used, this value won't matter for the function calls within the calling functions.

Examples

The following table shows 3 different ways of performing the same Change Signature refactoring.

In all the cases, the function `result()` is renamed to `generate_result()` and a new parameter `input` is added to this function.

The examples show how the function call, the calling function (`show_result()`) and other code fragments may be affected depending on the refactoring settings.

Before After

```
// This is the function whose signature will be changed:
function doSomething($a) {
// some code here
}

function refactor($a)
{

// Here is the function call:
$this->doSomething($a);
}

// Now we'll rename the function and
// add one parameter.
```

```
// The function has been renamed to doSomeRefactoring.
// The new parameter $b has been added.

function doSomeRefactoring($a,$b) {
// some code here
}

function refactor($a,$b) {

// The function call has been changed accordingly:
$this->doSomeRefactoring($a,$b);
}
```

```
// This is the function whose signature will be changed:

function doSomething($a) {
// some code here
}

function refactor($a)
```

```
// The function has been renamed to doSomeRefactoring.
// The new parameter $b has been added.

function doSomeRefactoring($a,$b) {
// some code here
}

function refactor($a)
```

```
// This is the function whose signature will be changed:

function doSomething($a) {
// some code here
}


function refactor($a)
{

// Here is the function call:
$this->doSomething($a);
}

// Now we'll rename the function and add one parameter.
//We will also propagate this new parameter
//through the calling function refactor() to the function call.
```

```
// The function has been renamed to doSomeRefactoring.
// The new parameter $b has been added.

function doSomeRefactoring($a,$b) {
// some code here
}

// Note the new function parameter:
function refactor($a,$b) {

// The function call has been changed accordingly:
$this->doSomeRefactoring($a,$b);
}
```

Changing a function signature

1. In the editor, place the cursor within the name of the function whose signature you want to change.
2. Do one of the following:
   - Press `Ctrl+F6`.
   - Choose Refactor | Change Signature on the main menu or .
   - Choose Refactor | Change Signature on the context menu.
3. In the Change Signature dialog, make the necessary changes to the function signature and specify what other, related, changes are required.
   You can:
   - Change the function name. To do that, edit the text in the Name field.
   - Manage the function parameters using the table and the buttons in the Parameters area:
     - To add a new parameter, click + and specify the properties of the new parameter in the corresponding table row.
       When adding parameters, you may want to propagate these parameters to the functions that call the current function.

       In the PHP context, when the Change signature refactoring is invoked from the constructor of a class, the new parameter can be initialized as a class field. To do that, use the Create and initialize class properties check box:
       - When this check box is selected and you add a parameter, this parameter is initialized as a field. PhpStorm creates a protected field with the same name as this parameter and adds a line with the following assignment: `$this-><parameter_name> = $<parameter_name>;`
       - When the check box is cleared, a parameter is added without initialization.

       For example, you have the following constructor:

```
class ChangeSignatureNewParam {
function __construct() {
$a = "Constructor in ChangeSignatureNewParam";
print $a;
}
}
```

If you invoke the Change signature refactoring from the __construct() method and add a new `$q` parameter, the result will depend on whether you select or clear the Create and initialize class properties check box:

The Create and initialize class properties **check box is selected**	**The** Create and initialize class properties **check box is cleared**

`class ChangeSignatureNewParam {`
`private $q;`
`function __construct( $q ) {`
`$a = "Constructor in ChangeSignatureNewParam";`
`print $a;`
`$this->q = $q;`
`}`
`}`
	`class ChangeSignatureNewParam {`
	`function __construct( $q ) {`
	`$a = "Constructor in ChangeSignatureNewParam";`
	`print $a;`
	`}`
	`}`

     - To remove a parameter, click any of the cells in the corresponding row and click —.
     - To reorder the parameters, use the ↑ and ↓ buttons. For example, if you wanted to put a certain parameter first in the list, you would click any of the cells in the row

corresponding to that parameter, and then click the required number of times.

- To change the name or the default value of a parameter, make the necessary updates in the table of parameters (in the fields Name and Default value respectively).

- Propagate new method parameters (if any) along the hierarchy of the functions that call the current function.
  (There may be functions that call the function whose signature you are changing. These functions, in their turn, may be called by other functions, and so on. You can propagate the changes you are making to the parameters of the current function through the hierarchy of calling functions and also specify which calling functions should be affected and which shouldn't.)

  To propagate a new parameter:

  1. Click the Propagate Parameters button .
  2. In the left-hand pane of the Select Methods to Propagate New Parameters dialog, expand the necessary nodes and select the check boxes next to the functions you want the new parameters to be propagated to.
     To help you select the necessary functions, the code for the calling function and the function being called is shown in the right-hand part of the dialog (in the Caller Method and Callee Method panes respectively).

     As you switch between the functions in the left-hand pane, the code in the right-hand pane changes accordingly.
  3. Click OK.

4. To perform the refactoring right away, click Refactor.
   To see the expected changes and make the necessary adjustments prior to actually performing the refactoring, click Preview.

Note Code completion is available in the Default value field of the table in the Parameters area.

Copy/Clone

The Copy refactoring allows you to copy a class, file, or directory with its entire structure from one directory to another, or clone it within the same directory.

To copy a class, file, or directory

1. Select the desired item in one of the views or open it in the editor.
2. Do one of the following:
   - On the main menu or on the context menu of the selection, choose Refactor | Copy.
   - Press `F5`.
   - Select the desired class in the Project tool window and drag it to the target destination with the `Ctrl` key pressed.
3. In the Copy dialog box that opens specify the new name and destination, then click OK.

To clone a class or file

1. Select the desired item in one of the views or open it in the editor.
2. Press `N/A`.
3. In the Clone dialog box that opens specify the new name, then click OK.

Extract Constant

The Extract Constant refactoring makes your source code easier to read and maintain. It also helps you avoid using hard-coded constants without any explanations about their values or purpose.

On this page:

- JavaScript example
- PHP examples
- How to extract a constant

JavaScript example

Before After

```
Parenizor.method('toString', function () {
                    return '(' + this.getValue() + ')';
                    }
                    )
```

```
Parenizor.method('toString', function () {
                    const string = '('
                    + this.getValue()
                    + ')';
                    return string;
                    }
                    )
```

PHP examples

- Extracting a class constant
- Extracting a constant outside a class

Extracting a class constant

When a constant is extracted within a class definition, the new constant is defined through the `const` keyword and referenced through the `self` keyword.

Before After

```
class const_extraction {
                    public static function find($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::execute($params['param_query']);
                    }
                    }

                    public static function findAll($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::executeAll($params['param_query']);
                    }
                    }
                    }
```

```
class const_extraction {
                    const PARAM_QUERY = 'param_query';

                    public static function find($params){
                    if (isset($params[self::PARAM_QUERY]))
                    {
                    $result = MyDatabase::execute($params[self::PARAM_QUERY]);
                    }
                    }

                    public static function findAll($params){
                    if(isset($params[self::PARAM_QUERY])){
                    $result = MyDatabase::executeAll($params[self::PARAM_QUERY]);
                    }
                    }
                    }
```

Extracting a constant outside a class

When a constant is extracted outside a class definition, you can choose whether it will be defined through the `const` keyword or through the `define()` function.

Before After

```
<?php
                    function find($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::execute($params['param_query']);
                    }
                    }

                    function findAll($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::executeAll($params['param_query']);
                    }
                    }
```

Const definition

```
<?php
                    const PARAM_QUERY = 'param_query';

                    function find($params){
                    if (isset($params[PARAM_QUERY])) {
                    $result = MyDatabase::execute($params[PARAM_QUERY]);
                    }
                    }

                    function findAll($params){
                    if (isset($params[PARAM_QUERY])) {
                    $result = MyDatabase::executeAll($params[PARAM_QUERY]);
                    }
                    }
```

Define() definition

```php
<?php
define('PARAM_QUERY',
'param_query');

function find($params){
if(isset($params[PARAM_QUERY])) {
$result = MyDatabase::execute($params[PARAM_QUERY]);
}
}

function findAll($params){
if(isset($params[PARAM_QUERY])) {
$result = MyDatabase::executeAll($params[PARAM_QUERY]);
}
}
```

To extract a constant

1. In the editor, position the cursor inside the expression to be replaced with a constant.
2. Choose Refactor | Extract Constant on the main menu or on the context menu of the selection. Alternatively press `Ctrl+Alt+C`.
3. If a constant can be extracted from several expressions in the current context, PhpStorm shows all the relevant expressions in a pop-up list. Select the expression to apply the refactoring to.
4. In the ExtractConstant dialog box, type the name of the new constant in the Name text box.
5. Specify the scope to apply refactoring in:
   - To have only the selected expression replaced, clear the Replace all occurrences check box.
   - To have PhpStorm replace the selected expression wherever it is used, select the Replace all occurrences check box.
6. If the refactoring is invoked outside a PHP class definition, choose the method to define the PHP constant in the Constant syntax area.
   - To have PhpStorm define the constant through the define() function , choose define.
   - To have PhpStorm define the constant through the `const` keyword outside a class definition, choose const.
     This method preserves the approach accepted in PHP version 5.3.0. Note that such constants are defined during compilation therefore they should be declared at the top-level scope but not inside functions, loops or if statements.

   Note If the refactoring is applied inside a class definition, it will result in extracting a class constant which is always defined through the `const` keyword by default, therefore the choice is irrelevant.
7. Click OK to start the refactoring.

Extract Field

The Extract Field refactoring declares a new field and initializes it with the selected expression. The original expression is replaced with the usage of the field.

– Example

– Extracting a field using the dialog box

Example

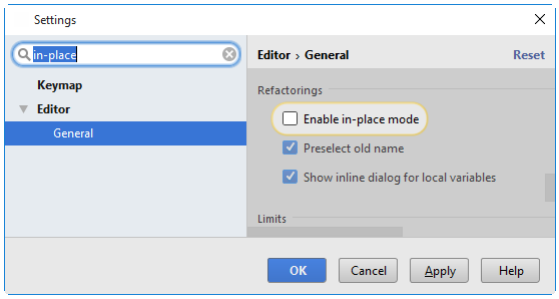
Before After

```
public static function
                    find($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::execute($params['param_query']);
                    }
                    }

                    public static function findAll($params){
                    if (isset($params['param_query'])) {
                    $result = MyDatabase::executeAll($params['param_query']);
                    }
                    }
```

```
public static $query
                    = 'param_query';

                    public static function find($params){
                    if (isset($params[self::$query]))
                    {
                    $result = MyDatabase::execute($params[self::$query]);
                    }
                    }

                    public static function findAll($params){
                    if(isset($params[self::$query])){
                    $result = MyDatabase::executeAll($params[self::$query]);
                    }
                    }
```

To extract a field using the dialog box

If the Enable in place refactorings check box is cleared on the Editor settings, the Extract Field refactoring is performed by means of the Extract Field Dialog .



1. In the editor, select the expression or variable to be replaced with a field, or just place the cursor within such an expression or variable declaration.
2. In the main menu, or the context menu of the selection, choose Refactor | Extract Field, or press `Ctrl+Alt+F`.
3. In the Expressions pop-up menu, select the expression to be replaced. Note that PhpStorm highlights the selected expression in the editor.
4. In the Extract Field Dialog that opens, specify the type and name of the new field.
5. To automatically replace all occurrences of the selected expression (if it is found more than once), select the option Replace all occurrences.
6. Click OK to create the field.

Extracting an interface

1. Select a class in the Project view or Structure view or select its name in the editor.

2. On the main menu or on the context menu of the selection, choose Refactor | Extract | Interface. The Extract Interface dialog box appears.

3. In the Interface Name text box, specify the name of the interface to be created. By default, PhpStorm suggests the name of the current class from which the interface will be extracted. Update the name otherwise PhpStorm will report an error and the refactoring will no start.

4. Specify whether you want PhpStorm to apply the extracted interface immediately:
   - To have PhpStorm create an interface and attempt to replace references to the source class with references to the interface in PhpDoc comments and parameters of methods, select the Replace class reference with interface where possible check box. The proposed changes are compulsory displayed in a Refactoring Preview pane of the Find tool window. Note, that if an instance references a method or field, which are not defined in the interface, it won't be suggested for replacement.
   - To have PhpStorm just create an interface in a separate file and mark the source class as implementation of the new interface, clear the Replace class reference with interface where possible check box.

5. Specify the namespace to which the new interface will belong. By default, the namespace of the source class in selected.

6. In the Target destination directory drop-down list, specify the folder to store the dedicated file created for the new interface. By default, the field is read-only and shows the folder that corresponds to the namespace to which the interface will belong according to the PSR0 standard .
   To have the class with the interface stored in another folder, click F2 and specify the folder of your choice.

7. In the Members to form interface area, specify the methods to be included in the interface. This table shows all the methods detected in the current class. To have a method moved to the interface specified in the Move Member To Class, select the check box next to the method.

8. In the PhpDoc area, specify how you want PhpDoc comments treated. The available options are:
   - As is: select this option to have the documentation left where it is in the source class.
   - Copy: select this option to have the inline documentation copied to the extracted interface without removing it from its current location.
   - Move: select this option to have the inline documentation moved to the extracted interface and deleted from its current location in the source class.

9. Click Refactor when ready. If PhpStorm shows you a Refactoring Preview in the Find tool window, review the suggested changes. To have the interface extracted and the proposed changes applied, click Do Refactor.

Extract Method

- Basics
- Examples
  - JavaScript example
  - PHP Extract Method example
  - PHP Extract Function example
- Extracting a JavaScript or PHP function

Basics

When the Extract Method refactoring is invoked , PhpStorm analyses the selected block of code and detects variables that are the input for the selected code fragment and the variables that are output for it.

The detected output variable is used as a return value for the extracted function.

In the JavaScript context, this refactoring always results in a function.

In the PHP context, the result of applying the Extract Method refactoring depends on the location of the selected code fragment.

- If the selection is made inside a method of a class, the refactoring extracts a method. This case is applicable when you are using PHP 5.0 and higher.
- If the selection is made inside a function or a script, the refactoring extracts a function.

Examples

JavaScript example

Before After

```
function multiplication(a,b) {
c = a + b;
d = c * c;
return d;
}
```

```
function sum(a,b);
return a + b;

function multiplication(a,b) {
c = sum(a,b);
d = c * c;
return d;
}
```

PHP Extract Method example

Before After

```
public function init()
{
$this->_router = $this->getFrontController()->getRouter();
}
```

```
public function init()
{
$this->_router = $this->getRouter();
}

public function getRouter()
{
return $this->getFrontController()->getRouter();
}
```

PHP Extract Function example

Before After

```
<?php
if ('POST' != $_SERVER['REQUEST_METHOD'])
{
header('Allow: POST');
header('HTTP/1.1 405 Method Not Allowed');
header('Content-Type: text/plain');
exit;
}
?>
```

```
<?php
function printEmptyHeader()
{
header('Allow: POST');
header('HTTP/1.1 405 Method Not Allowed');
header('Content-Type: text/plain');
}

if ('POST' != $_SERVER['REQUEST_METHOD'])
{
printEmptyHeader();
exit;
}
?>
```

Extracting a JavaScript or PHP function

To extract a function in the JavaScript context

1. In the editor, select a block of code to be transformed into a function.

   Tip The selected code fragment does not necessarily have to be a set of statements. It may also be an expression used somewhere in the code.

2. On the main menu or on the context menu of the selection, choose Refactor | Extract Method or press `Ctrl+Alt+M`.

3. In the Extract Function dialog box that opens, specify the name of the new function.

4. To have the new function defined through a function expression, select the Declare functional expression check box.

5. In the Parameters area, configure the set of variables to be passed to the new function as parameters. By default, all the variables from the specified scope are listed.
   - To have a variable included in the parameter set, select the check box next to it.
   - To change the order of parameters, use the Move Up and Move Down buttons.

6. View and check the declaration of the function to be generated in the Signature preview read-only area.

To extract a function or method in the PHP context

1. In the editor, select a block of code to be transformed into a function or method.

   Tip The selected code fragment does not necessarily have to be a set of statements. It may also be an expression used somewhere in the code.

2. On the main menu or on the context menu of the selection, choose Refactor | Extract Method or press `Ctrl+Alt+M`.
   The title of the dialog box that opens is either Extract Method or Extract Function depending on the intended refactoring output .

3. Specify the name of the new function or method.

4. If a method is to be extracted, choose the relevant access modifier for it in the Visibility area:
   - Public
   - Protected
   - Private

5. Configure the output of the new method or function in the Return output variable(s) through area.
   The Output variable(s) read-only field displays all the variables that make up the output of the selected code fragment. Specify the way in which the new method or function will return these variables to the callee.
   - To have the output variables returned by value, select the Return statement option. The result of this choice depends on the number of detected output variables. If there is exactly one output variable, it will be used as the return value. If the selection outputs several variables, these variables will be returned as an array.
   - To have the output variables returned by reference, select the Parameter(s) passed by reference option. In this case, no return statement will be generated. Instead, all the detected output variables will be added to the list of input parameters . Their names will be prepended with an ampersand `&` both in the Parameters list and in the declaration of the new method/function, as shown in the Signature preview read-only area.

6. In the Parameters area, configure the input for the new function or method.
   Originally, the list contains the variables that PhpStorm has detected as the input for the new method or function. However, if you have chosen to return the new method/function output by reference , the list also contains all the output variables.

   To change the order of parameters, use the Move Up and Move Down buttons. This order determines the order in which the parameters are listed in the declaration of the new method or function.

   If necessary, rename the desired parameters, by double-clicking the corresponding parameter lines and entering new names.

7. Optionally, to have PhpStorm transform tail break or continue statements if the selection contains any, select the Replace tail "break/continue" statements with return statements check box.

8. View and check the declaration of the function or method to be generated in the Signature preview read-only area.

Extract Parameter

The Extract Parameter refactoring is used to add a new parameter to a function declaration and to update the function calls accordingly.

- Extracting a Parameter in PHP
- JavaScript Example

See also Change Signature .

Extracting a Parameter in PHP

In the example below, a new parameter $c will be added to the Calculate() function to replace 10:

Before After

```php
<?php
class Class1 {
public function Calculate($i){
while ( $i < 10
) {
$i = $i + 1;
return $i;
};
}

public function DisplaySum(){
$a = 1;
$result = $this -> Calculate($a);
echo "The final result is " . $result;
}
}
```

```php
<?php
class Class1 {
public function Calculate($i,$c){
while ( $i < $c
) {
$i = $i + 1;
return $i;
};
}

public function DisplaySum(){
$a = 1;
$result = $this -> Calculate($a,10);
echo "The final result is " . $result;
}
}
```

1. In the editor, place the cursor within the expression to be replaced by a parameter.
2. Do one of the following:
   - Press `Ctrl+Alt+P`.
   - Choose Refactor | Extract | Parameter on the main menu.
   - Choose Refactor | Extract | Parameter from the context menu.
3. In the Introduce Parameter dialog box that opens, type the name of the new parameter.

When you click OK, the code is updated: the new parameter is added to the definition of the function and to its call.

JavaScript Example

Before After

A new parameter will be added to this function to replace the 1's:

```javascript
function calculate_sum(i) {
alert('Adding '
+ 1 + ' to
' + i);
return (1 + i);
}

function show_sum() {
alert('Result:
' + calculate_sum(5));
}
```

The new parameter i2 has been added as an optional parameter:

```javascript
function calculate_sum(i, i2) {
i2 = i2 || 1;
alert('Adding '
+ i2 + ' to
' + i);
return (i2 + i);
}

function show_sum() {
// The function call has not changed:
alert('Result:
' + calculate_sum(5));
}
```

A new parameter will be added to this function to replace the 1's. When adding a new parameter we'll specify that it should be a required one.

```javascript
function calculate_sum(i) {
alert('Adding '
+ 1 + ' to
' + i);
return (1 + i);
}

function show_sum() {
alert('Result:
' + calculate_sum(5));
}
```

The new parameter i2 has been added as a required parameter:

```javascript
function calculate_sum(i, i2) {
alert('Adding '
+ i2 + ' to
' + i);
return (i2 + i);
}

function show_sum() {
alert('Result:
' + calculate_sum(5, 1));
}
```

See Extract Parameter in JavaScript for details.

Extract Variable

- Basics
- PHP Example
- Extracting variable in-place
- Extracting variable with a dialog

Basics

The Extract Variable refactoring puts the result of the selected expression into a variable. It declares a new variable and uses the expression as an initializer. The original expression is replaced with the new variable (see the examples below).

To perform this refactoring, you can use:

- In-place refactoring . In this case you specify the new name right in the editor.
- Refactoring dialog , where you specify all the required information. To make such a dialog accessible, you have to clear the check box Enable in-place mode in the editor settings.

You can select the expression to be replaced with a variable yourself. You can as well use smart expression selection . In this case PhpStorm will help you select the desired expression.

This refactoring is also available for JavaScript and Sass .

PHP Example

Before After

```
public function getFeedObject($title, $description)
                    {
                    global $wgSitename, $wgContLanguageCode, $wgFeedClasses, $wgTitle;
                    if (!isset($wgFeedClasses[$this->format]))
                    return false;
                    return new $wgFeedClasses[$this->format]
                    ("$wgSitename - {$title} [$wgContLanguageCode]", htmlspecialchars());
                    }
```

```
public function getFeedObject($title, $description)
                    {
                    global $wgSitename, $wgContLanguageCode, $wgFeedClasses, $w
                    $feedTitle = "$wgSitename - {$title} [$wgContLanguageCode]"
                    if (!isset($wgFeedClasses[$this->format]))
                    return false;
                    return new $wgFeedClasses[$this->format]
                    ($feedTitle, htmlspecialchars());
                    }
```

Extracting variable in-place

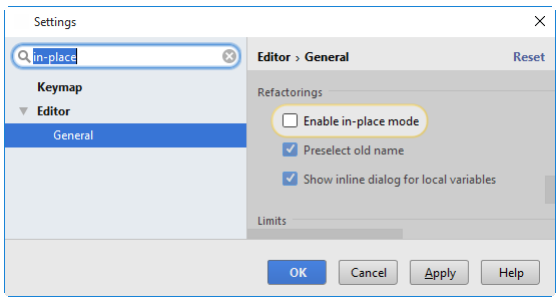
To extract a variable using in-place refactoring, follow these steps

1. In the editor, select the expression to be replaced with a variable. You can do that yourself or use the smart expression selection feature to let PhpStorm help you. So, do one of the following:
   - Highlight the expression. Then choose Refactor | Extract | Variable on the main menu or on the context menu. Alternatively, press `Ctrl+Alt+V`.
   - Place the cursor before or within the expression. Choose Refactor | Extract Variable on the main menu or on the context menu. or press `Ctrl+Alt+V`.
2. If more than one occurrence of the selected expression is found, select Replace this occurrence only or Replace all occurrences in the Multiple occurrences found pop-up menu. To select the required option, just click it. Alternatively, use the Up and Down arrow keys to navigate to the option of interest, and press `Enter` to select it.
3. Specify the name of the variable. Do one of the following:
   - Select one of the suggested names from the pop-up list. To do that, double-click the suitable name. Alternatively, use the Up and Down arrow keys to navigate to the name of interest, and `Enter` to select it.
   - Edit the name by typing. The name is shown in the box with red borders and changes as you type. When finished, press `Enter`.

Extracting variable with a dialog

To extract a variable using the dialog box

If the Enable in place refactorings check box is cleared in the Editor settings, the Extract Variable refactoring is performed by means of the dialog box.



1. In the editor, select the expression to be replaced with a variable. You can do that yourself or use the smart expression selection feature to let PhpStorm help you. So, do one of the following:
   - Highlight the expression. Then choose Refactor | Extract | Variable on the main menu or on the context menu. Alternatively, press `Ctrl+Alt+V`.
   - Place the cursor before or within the expression. Choose Refactor | Extract Variable on the main menu or on the context menu. or press `Ctrl+Alt+V`. In the Expressions pop-up menu, select the expression. To do that, click the required expression. Alternatively, use the Up and Down arrow keys to

navigate to the expression of interest, and then press `Enter` to select it.

> **Note** The Expressions pop-up menu contains all the expressions appropriate for the current cursor position in the editor.
>
> When you navigate through the suggested expressions in the pop-up, the code highlighting in the editor changes accordingly.

2. In the Extract Variable Dialog dialog:
   1. Specify the variable name next to Name. You can select one of the suggested names from the list or type the name in the Name box.
   2. If more than one occurrence of the selected expression is found, you can select to replace all the found occurrences by selecting the corresponding check box. If you want to replace only the current occurrence, clear the Replace all occurrences check box.
   3. For ActionScript, you can choose to introduce a constant rather than a variable. To do that, select the Make constant check box.
   4. Click OK.

Refer to the following sections for the other language- and framework-specific refactorings:

- For the JavaScript procedure, refer to the section Extract Variable in JavaScript .
- For the Sass procedure, refer to the section Extract Variable for Sass .

Extract Variable for Sass

You can replace a Sass expression with a local or global variable.

To perform this refactoring, you can use:

- In-place refactoring . In this case you specify the new name right in the editor.
- Refactoring dialog , where you specify all the required information. To make such a dialog accessible, you have to clear the check box Enable in-place mode in the editor settings.

Example

Before After

```
$blue: #3bbfce
$margin: 16px

.border
padding: $margin / 2
margin: $margin / 2
border-color: $blue
```

```
$blue: #3bbfce
$margin: 16px;
$var: $margin / 2

.border
padding: $var
margin: $var
border-color: $blue
```

To extract a variable using in-place refactoring

1. In the editor, select the expression to be replaced with a variable. You can do that yourself or use the smart expression selection feature to let PhpStorm help you. So, do one of the following:
   - Highlight the expression. Then choose Refactor | Extract | Variable on the main menu or on the context menu. Alternatively, press `Ctrl+Alt+V`.
   - Place the cursor before or within the expression. Choose Refactor | Extract Variable on the main menu or on the context menu. or press `Ctrl+Alt+V`. In the Expressions pop-up menu, select the expression. To do that, click the required expression. Alternatively, use the Up and Down arrow keys to navigate to the expression of interest, and then press `Enter` to select it.

   Note The Expressions pop-up menu contains all the expressions appropriate for the current cursor position in the editor.

   When you navigate through the suggested expressions in the pop-up, the code highlighting in the editor changes accordingly.

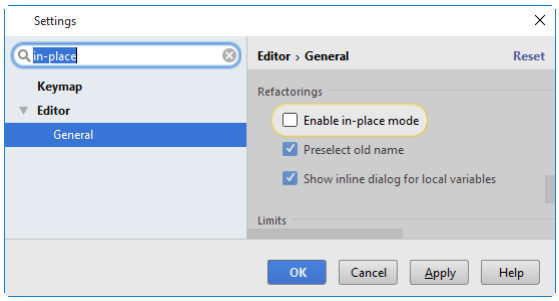
2. If more than one occurrence of the selected expression is found, select Replace this occurrence only or Replace all occurrences in the Multiple occurrences found pop-up menu. To select the required option, just click it. Alternatively, use the `Up` and `Down` arrow keys to navigate to the option of interest, and press `Enter` to select it.

3. Select the place in the source code, where the new variable will be declared. The declaration can be global (a variable is available throughout the whole file), or local (a variable is declared immediately before use, and is available in the current block only).

4. Specify the name of the variable. Do one of the following:
   - Select one of the suggested names from the pop-up list. To do that, double-click the suitable name. Alternatively, use the `Up` and `Down` arrow keys to navigate to the name of interest, and `Enter` to select it. When finished, press `Escape`.
   - Edit the name by typing. The name is shown in the box with red borders and changes as you type. When finished, press `Escape`.

To extract a variable using the dialog box

If the Enable in place refactorings check box is cleared in the Editor settings, the Extract Variable refactoring is performed by means of the dialog box.



1. Select the desired expression, and invoke Extract Variable refactoring as described above .

2. If more than one expression is detected for the current cursor position, the Expressions list appears. If this is the case, select the required expression. To do that, click the expression. Alternatively, use the `Up` and `Down` arrow keys to navigate to the expression of interest, and then press `Enter` to select it.

3. In the Extract Variable dialog for Sass :
   - Specify the variable name. You can select one of the suggested names from the list or type the name in the Name field.
   - Specify the place for declaration. Select the desired place (global or local) from the drop-down list.
   - If more than one occurrence of the selected expression is found, you can select to replace all the found occurrences by selecting the corresponding check box. If you want to replace only the current occurrence, clear the Replace all occurrences check box.
   - Click OK.

```
var browserName = "N/A";
if (navigator.appName.indexOf("
    browserName = "NS";
} else if (navigator.appName.in
    browserName = "MSIE";
} else if (navigator.appName.in
    browserName = "O";
}

var browserVersion = "0";
```

Name: appName

Replace all occurrences (3 occurrences found)

Var kind

- Introduce var
- Introduce constant
- Introduce local variable (JavaScript 1.7+)

Inline

PhpStorm provides the following inline refactorings:

- The Inline Variable refactoring replaces redundant variable usage with its initializer. This refactoring is opposite to Extract Variable .
- The Inline Function/Method refactoring results in placing the method's or function's body into the body of its caller(s); the method or function is deleted. This refactoring is opposite Extract Method .

Inline Variable

PHP Example

Before After

```
public function getFeedObject($title, $description)
                    {
                    global $wgSitename, $wgContLanguageCode, $wgFeedClasses, $wgTitle;
                    $feedTitle = "$wgSitename - {$title} [$wgContLanguageCode]";
                    if (!isset($wgFeedClasses[$this->format]))
                    return false;
                    return new $wgFeedClasses[$this->format]
                    ($feedTitle, htmlspecialchars());
                    }
```

```
public function getFeedObject($title, $description)
                    {
                    global $wgSitename, $wgContLanguageCode, $wgFeedClasses, $wgTi
                    if (!isset($wgFeedClasses[$this->format]))
                    return false;
                    return new $wgFeedClasses[$this->format]
                    ("$wgSitename - {$title} [$wgContLanguageCode]", htmlspecialch
                    }
```

JavaScript Example

Before After

```
Parenizor.method('toString', function ()
                    {
                    var string = '(' + this.getValue()
                    + ')';
                    return string;
                    }
```

```
Parenizor.method('toString', function ()
                    {
                    return '(' + this.getValue() + ')';
                    }
```

Inline Method or Function

Before After

```
function sum(a,
                    b){
                    return a + b;
                    }

                    function multiplication(a, b){
                    c = sum(a, b);
                    d = c * c;
                    return d;
                    }

                    function division(a,
                    b){
                    result = sum(a, b) / multiplication(a, b);
                    return result;
                    }
```

```
function multiplication(a, b){
                    c = a + b;
                    d = c * c;
                    return d;
                    }

                    function division(a,
                    b){
                    result = a + b
                    / multiplication(a,
                    b);
                    return result;
                    }
```

To perform the inline refactoring

1. Place the caret in the editor at the desired symbol to be inlined.
2. Do one of the following:
   - On the main menu or on the context menu, choose Refactor | Inline.
   - Press `Ctrl+Alt+N`.
3. In the Inline dialog box that corresponds to the selected symbol, confirm the inline refactoring.

Make Method Static

The Make Method Static refactoring converts an instance method to a static one and automatically corrects all calls, implementations and overridings of the method.

Performing the Refactoring

1. Select the method to be refactored in the Structure view, or right-click the method name in the editor. On the main menu, or on the context menu of the selection, choose Refactor | Make Static. If your method does not need any parameters, nor rely on accessing properties or methods of this object, you will get the static method made silently, without displaying any dialog boxes:

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    static private function getFormattedDate() {
                    $format = getSettings()[ 'dateFormat' ];
                    return time( $format );
                    }
                    }
```

For more complicated cases, the Make Method Static dialog box opens.

2. If you currently use the existing object via `this`, select the Add object as a parameter with name checkbox to pass in an instance of the object via a parameter. In the text box below, specify the name of the parameter to be generated. After the refactoring, the new parameter will be documented in the PHPDoc block.

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = $this->getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    /**
                        * @param MyClass $instance
                        * @return
                        */
                    static private function getFormattedDate($instance) {
                    $format = $instance->getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

3. If you access the properties of the class, use the Add parameters for fields area to pass the value of the property as a parameter instead of accessing the object inside the newly created static method.
   1. Select the Add parameters for fields check box.
   2. In the Parameters list, which shows all the possibly suggested parameters, select the check boxes next to the ones that you want to pass the values in.

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = $this->timeFormat;
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    /**
                        * @param $timeFormat
                        * @return int;
                        */
                    static private function getFormattedDate($timeFormat) {
                    $format = timeFormat;
                    return time($format);
                    }
                    }
```

4. To preview the results, click Preview and examine the reuslt of the refactoring in the Find tool window . Apply the changes, if no issues arizs.

Move Refactorings

On this page:

- Move refactorings: basics
- Moving a file or a directory to another directory
- Moving a PHP namespace
- Moving a static PHP method, variable, or constant to another class
- Moving a class to another namespace
- Examples of moving PHP classes
  - Moving a Class to a New Namespace under the Current Namespace
  - Moving a Class between Two Namespaces under the Same Parent
  - Moving a Class to an Existing Namespace under Another Parent Namespace
  - Moving a Class to a New Namespace Outside the Immediate Parent Namespace

Move refactorings: basics

The Move refactorings allow you to move classes, static members (methods, variables, and constants), files and directories within a project. So doing, PhpStorm automatically corrects all references to the moved symbols in the source code.

The following Move refactorings are available:

- The Move File refactoring moves a file to another directory.
- The Move Directory refactoring moves a directory to another directory.
- The Move Class refactoring moves a PHP class to another existing namespace or creates a new one and moves the class to it.
- The Move Static Member refactoring moves a static method, variable, or constant from one class to another.

Moving a file or a directory to another directory

To move a file or directory to another directory, follow these steps:

1. Select the file or directory in the Project tool window.
2. Do one of the following:
   - On the main menu, choose Refactor, and then choose Move.
   - On the context menu of the selection, choose Refactor, and then choose Move.
   - Press `F6`.

   The dialog that opens depends on the type of the selected symbol.
3. In the To directory field, specify the folder to move the selected file or folder to. Choose an existing folder from the drop-down list or type the full path to the parent folder to be created.
4. To have the references to the selected file or folder updated according to the refactoring result, select the Search for references check box.

Moving a PHP namespace

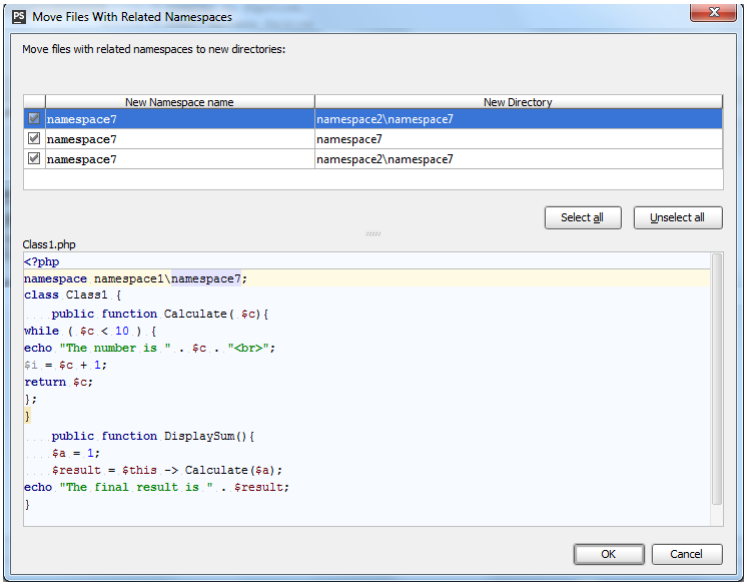
PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when moving namespaces.

When you specify a namespace to move a namespace to, PhpStorm automatically updates the Target Destination Directory field, which shows the path to the folder that corresponds to the namespace in question.

To move a namespace with all or some of its related classes to another namespace:

1. In the editor, place the caret at the namespace to move and choose Refactor | Move on the main menu or on the context menu of the selection. Note that if you select the folder that corresponds to the required namespace in the Project tool window and then choose Refactor | Move, the Move Directory refactoring will be invoked.
2. In the Move Namespace Dialog that opens, specify the new namespace name and the folder that will correspond to it.
   1. The New Namespace Name field shows the fully qualified name of the selected namespace. Specify the new namespace name. Use only backslashes (`\`) as namespace separators.
   2. When the dialog box opens, the Target Destination Directory field shows the path to the folder which corresponds to the current namespace.
      The path is displayed in the format `...\<project root folder>\<path to the current namespace folder relative to the project root>`. The path is updated automatically as you specify the new namespace name. However, if you are going to move a namespace to another parent namespace, PhpStorm will not suggest the proper folder unless you appoint a root folder for your namespace structure by marking the relevant folder as Sources on the Directories page of the Settings dialog box, see Directories for details.

      Do one of the following:
      - Accept the preselected path displayed in the field.
      - Choose another path from the list. All of them are evaluated from the namespace root or from the current directory, so it is safe to choose any of them.
      - Click ... and select a folder in the dialog box that opens.
      - Press `F2` and edit the preselected path. Keep in mind that this may cause problems with automatic loading in the future.
3. Click Refactor and in the Move Files with Related Namespaces dialog box that opens specify the classes and files to be moved to the new namespace and the new folder.
   The upper pane of the dialog box lists the destination namespaces and folders for classes and files related to the namespace in question. Each item in the list corresponds to a class/file. When you move the cursor to an item, the bottom pane shows the contents of the file related with it.



- To have a class and the corresponding file moved to the destination namespace and the destination folder, select the check box next to the namespace/folder.
- To add all the items to the list or remove all of them from the list, click Select All or Unselect All respectively.

4. Click OK to start the refactoring.

Moving a static PHP method, variable, or constant to another class

To move a static PHP method, variable, or constant to another class, follow these steps:

1. Open a PHP class in the editor and select a static method, variable, or constant or just position the cursor at it.
2. On the context menu of the selection or on the main menu, choose Refactor | Move.
3. In the Move Static Member Dialog that opens, specify the following:
   1. In the Members to be moved to another class field, select the check boxes next to the methods, constants, and variables that you want to move to another class. The list shows all the static methods, variables, and constants detected in the current class.
   2. In the Move Member To Class text box, type the fully qualified name of the class where you want to move the members selected in the Members to be moved to another class list. PhpStorm provides completion for the class and namespace reference: as you type a class or namespace name, a suggestion list with the relevant item is displayed. You can also get a full list of available classes by pressing `Ctrl+Space`.

Moving a class to another namespace

PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when moving classes among namespaces.

When you specify a namespace to move a class to, PhpStorm automatically updates the Target Destination Directory field, which shows the path to the folder where the file that implements the class in question will be stored.

However, if you are going to move a class to a non-existing namespace under another parent namespace, PhpStorm will not suggest the proper folder unless you appoint a root folder for your namespace structure by marking the relevant folder as Sources on the Directories page of the Settings dialog box, see Directories for details.

To move a class to another namespace, follow these steps:

1. In the editor, place the caret at the class to move.
2. Do one of the following:
   - On the main menu, choose Refactor, and then choose Move.
   - On the context menu of the selection, choose Refactor, and then choose Move.
   - Press `F6`.
3. In the Move Class dialog box, that opens, specify the namespace to move the class to and the folder where the file that implements the class will be stored.
   1. In the Move Class <class name> to Namespace field, appoint the namespace to move the selected class to. This can be:
      - A new namespace under the current namespace.
      - Another subordinate namespace under the current parent.
      - An existing or a new namespace under another parent.

      In any case, the import statements referencing the class are updated accordingly.
   2. In the Target Destination Directory field, appoint the folder where you want PhpStorm to move the file that implements the class.
      PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when moving classes among namespaces.

      When you specify a namespace to move a class to, PhpStorm automatically updates the Target Destination Directory field, which shows the path to the folder file that implements the class in question.

      When the dialog box opens, the field shows the path to the folder which corresponds to the current namespace.

      The path is displayed in the format `...\<project root folder>\<path to the current namespace folder relative to the project root>`. The path is updated automatically as you specify the new namespace name. However, if you are going to move a namespace to another parent namespace, PhpStorm will not suggest the proper folder unless you appoint a root folder for your namespace structure by marking the relevant folder as Sources on the Directories page of the Settings dialog box, see Directories for details.

      Do one of the following:

- Accept the preselected path displayed in the field.
- Choose another path from the list. All of them are evaluated from the namespace root or from the current directory, so it is safe to choose any of them.
- Click ... and select a folder in the dialog box that opens.
- Press F2 and edit the preselected path. Keep in mind that this may cause problems with automatic loading in the future.

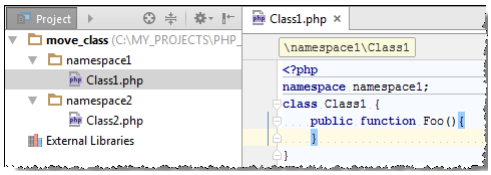
3. The Move class to a new file check box is selected. Leave this default selection.

4. Initiate refactoring by clicking Refactor or Preview, for more details, see Refactoring Source Code .

Examples of moving PHP classes

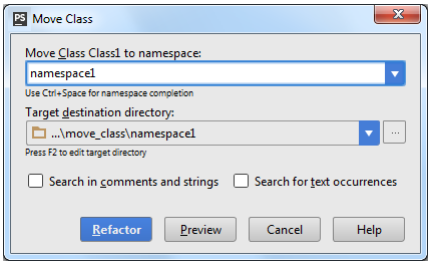
Moving a Class to a New Namespace under the Current Namespace

Suppose, your project `move_class` has a namespace `namespace1` with one class `Class1` implemented in the file `Class1.php`.

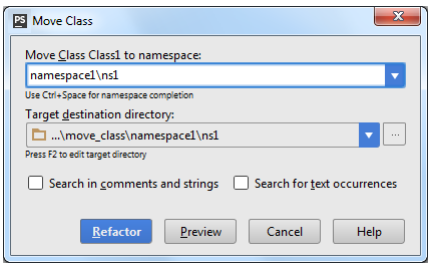


Let's create a subordinate namespace `namespace1\ns1` and move `Class1` to it.

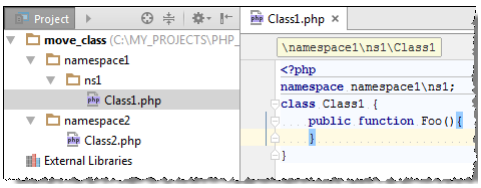
1. Open `Class1.php` in the editor, position the cursor at `Class1`, and press F6. The Move Class dialog box opens.



2. The Move class Class1 to namespace field shows the current namespace `namespace1`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace1` to `...\move_class\namespace1\ns1`.

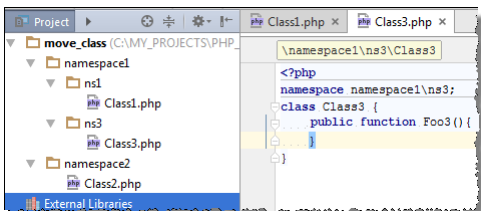


4. Press Refactor. As a result, PhpStorm creates a new subfolder `ns1` and moves the file `Class1` to it. Accordingly, the class `Class1` is moved to the `namespace1\ns1` namespace:



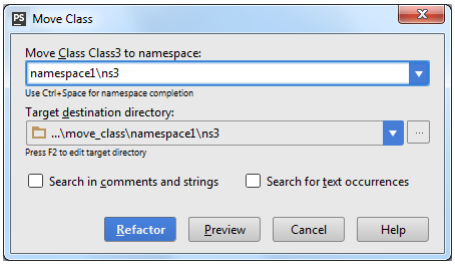
Moving a Class between Two Namespaces under the Same Parent

Suppose, your project `move_class` has a namespace `namespace1` with two subordinate namespaces `namespace1\ns1` and `namespace1\ns3`. These subordinate namespaces have `Class1` `Class3` respectively. In accordance with the PSR0 standard, these classes are implemented in the files `Class1.php` and `Class3.php` that are stored in the folders `...\move_class\namespace1\ns1` and `...\move_class\namespace1\ns3` respectively.

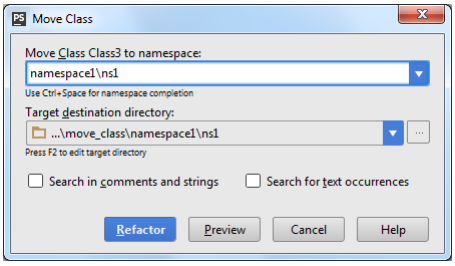


Let's move `Class3` from `namespace1\ns3` to `namespace1\ns1`.

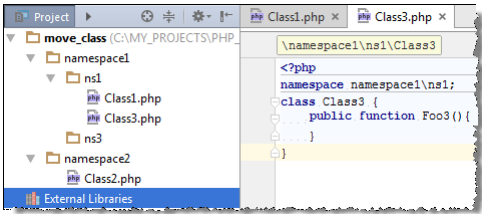
1. Open `Class3.php` in the editor, position the cursor at `Class3`, and press F6. The Move Class dialog box opens.



2. The Move class Class3 to namespace field shows the current namespace `namespace1\ns3`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace1\ns3` to `...\move_class\namespace1\ns1`.

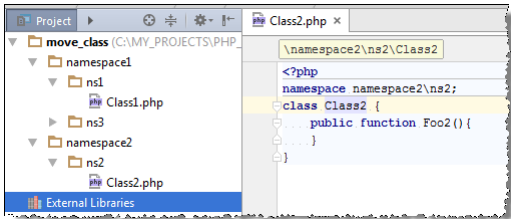


4. Press Refactor. As a result, PhpStorm moves the class `Class3` from `namespace1\ns3` to `namespace1\ns1`. Accordingly, the file `Class3` is moved from folder `...\move_class\namespace1\ns3` to folder `...\move_class\namespace1\ns1`:



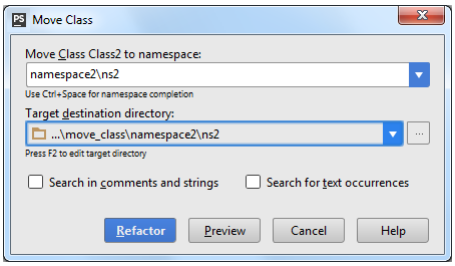
Moving a Class to an Existing Namespace under Another Parent Namespace

Suppose, your project `move_class` has two parent namespaces `namespace1` and `namespace2` with subordinate namespaces `namespace1\ns1` and `namespace2\ns2`. `Class2` is defined in `namespace2\ns2`, and in accordance with the PSR0 standard the file `Class2.php` that implements it is stored in `...\move_class\namespace2\ns2`.

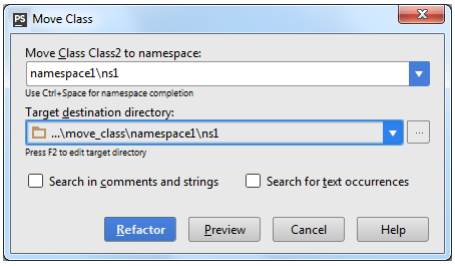


Let's move `Class2` from `namespace2\ns2` to `namespace1\ns1`.

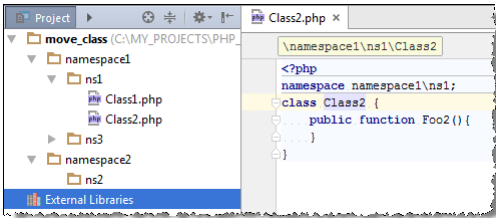
1. Open `Class2.php` in the editor, position the cursor at `Class2`, and press F6. The Move Class dialog box opens.



2. The Move class Class2 to namespace field shows the current namespace `namespace2\ns2`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace2\ns2` to `...\move_class\namespace1\ns1`.

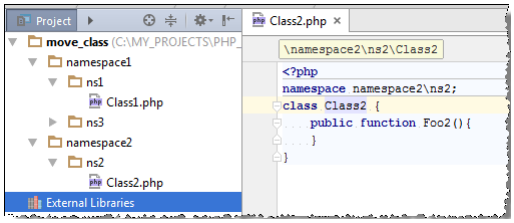


4. Press Refactor. As a result, PhpStorm moves the class `Class2` from `namespace2\ns2` to `namespace1\ns1`. Accordingly, the file `Class2` is moved from folder `...\move_class\namespace2\ns2` to folder `...\move_class\namespace1\ns1`:



Moving a Class to a New Namespace Outside the Immediate Parent Namespace

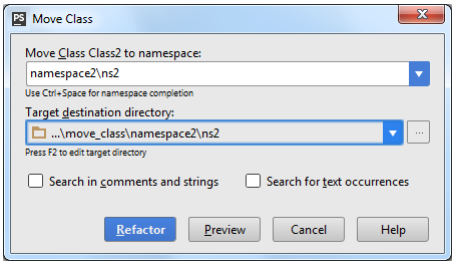
Suppose, your project `move_class` has two parent namespaces `namespace1` and `namespace2` with subordinate namespaces `namespace1\ns1` and `namespace2\ns2`. `Class2` is defined in `namespace2\ns2`, and in accordance with the PSR0 standard the file `Class2.php` that implements it is stored in `...\move_class\namespace2\ns2`.



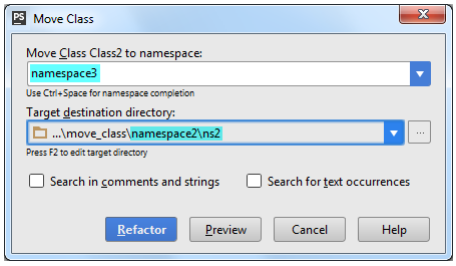
Let's move `Class2` from `namespace2\ns2` to a new namespace `namespace3` that does not exist yet. To meet the PSR0 requirements, a new folder `...\move_class\namespace3` for the `namespace3` namespace should be also created.

When starting the refactoring we should keep in mind that the new namespace is to be created outside the immediate parent namespace of our class's namespace, therefore PhpStorm will not suggest a proper folder for it. So we can either specify the target folder manually or appoint a root folder for namespaces which will be the starting point for PhpStorm to calculate the folder for the new namespace.

1. Open `Class2.php` in the editor, position the cursor at `Class2`, and press `F6`. The Move Class dialog box opens.



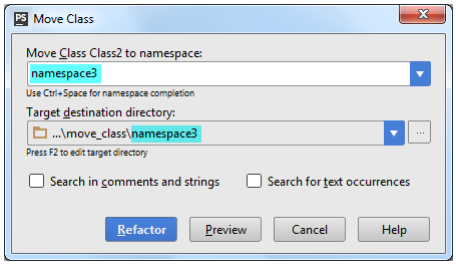
2. The Move class Class2 to namespace field shows the current namespace `namespace2\ns2`. Change it to `namespace3`. The result depends on whether you have appointed the root folder for your namespaces or not.
   - If no root folder is specified, the Target destination directory field still shows the path to `...\move_class\namespace2\ns2` because PhpStorm cannot calculate the path to the folder for the new namespace `namespace3` to be created:



   To solve the problem, press `F2` and type the path manually:

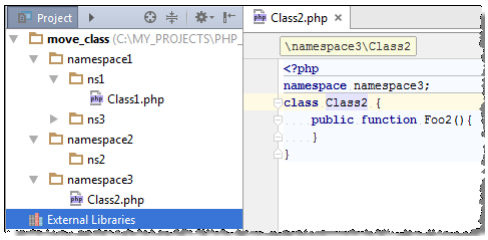
```
...\move_class\namespace2\ns2
```

   - Alternatively, specify the root folder for your namespaces by marking the `...\move_class\` folder as Sources as described in Configuring Folders Within a Content Root. Note, that this should be done before you start the refactoring, so if you have already opened the Move Class dialog box, close it first.
     After you have appointed the `...\move_class\` folder as the root for your namespaces, PhpStorm will calculate the folder for the new namespace and update the Target Destination Directory field automatically:



   - Press Refactor. As a result, PhpStorm moves the class `Class2` from `namespace2\ns2` to `namespace3`. Accordingly, the file `Class2` is moved from folder

`...\move_class\namespace2\ns2` to folder `...\move_class\namespace3` :



Pull Members Up

– Basics

– Example

– Pulling members up

Basics

The Pull Members Up refactoring allows you to move class members to a superclass .

Example

Before After

Pulling members up

To pull members up

1. Select the class to be moved to a superclass.
2. On the main menu or on the context menu, choose Refactor | Pull Members Up. The Pull Members Up dialog box appears.
3. Select the destination object (superclass).
4. In the Members section, select the members you want to move.
5. To move a method as abstract, select the check box in the column Make abstract next to the method in question.
6. Click Refactor to pull the selected members to their destination.

Push Members Down

The Push Members Down refactoring helps clean up the class hierarchy by moving class members to a subclass. The members are then relocated into the direct subclasses only.

Example

Before After

To push members down

1. In the editor open a class, whose members you need to push down.
2. On the main menu or on the context menu, choose Refactor | Push Members Down. Push Members Down dialog box displays the list of members to be pushed down.
3. In the Members to be pushed down area, select the members you want to move. Note that the member at caret is already selected.
   If pushing a member might cause problems, you will be notified with red highlighting. It means that, if the situation is unattended, an error will emerge after refactoring. PhpStorm prompts you with a Problems Detected dialog, where you can opt to ignore or fix the problem.
4. Preview and apply changes.

Rename Refactorings

On this page:

- Basics
- Renaming a symbol
- Renaming a file or directory

Basics

Rename refactorings allow you to rename symbols with all the references to them in the code corrected automatically.

The following rename refactorings are available in PhpStorm:

- Rename Class. The following usages are renamed:
  - Import statements
  - Qualified names of classes
- Rename Method. The following usages are renamed:
  - All calls of the method.
  - All overridden/implemented methods in subclasses.
- Rename Field.
- Rename Function.
- Rename Variable.
- Rename Parameter. The following usages are renamed:
  - All usages of the parameter.
  - The corresponding `param` tag in documentation comment.
- Rename CSS color value.
- Rename File.
- Rename Directory.

Renaming a symbol

To rename a symbol, follow these general steps

1. Select the item to be renamed.
   - To select a file, click the desired file in the Project tool window .
   - To select a symbol in the editor, place the caret at the name of the symbol to be renamed.
   - To select a symbol in the Project tool window , make sure that the members are shown, and then click the desired symbol.
   - To select a symbol in the Structure view , click the desired symbol in the Structure tool window.
2. Choose Refactor | Rename on the main menu or on the context menu, or press `Shift+F6`.
3. The subsequent behavior depends on the check box Enable in-place mode (Settings | Editor).
   - If this check box is selected, the suggested name appears right below the symbol in question. You can either accept suggestion, or type a new name. However, if you press `Shift+F6` once more, PhpStorm will display the Rename dialog box with more options.
   - If this check box is not selected, the Rename dialog box opens immediately.

   The set of controls and their names depend on the type of the symbol to be renamed.
4. Preview and apply changes .

Renaming a file or directory

To rename a file or directory

1. Select a desired file in the Project tool window.
2. Choose Refactor | Rename on the main or context menu or press `Shift+F6`.
3. In Rename File dialog box that opens, specify the new file name. Select Search in comments and strings checkbox to let PhpStorm apply changes to comments and strings.
4. Press Preview to observe possible changes in Find Tool Window. Press Refactor to proceed.
   PhpStorm finds all the occurrences of the file name and changes them respectively.

Safe Delete

The Safe Delete refactoring lets you safely remove files and symbols from the source code.

To make sure that deletion is safe, PhpStorm looks for the usages of the symbol being deleted. If such usages are found, you can explore them and make the necessary corrections in your code before the symbol is actually deleted.

Performing the refactoring

1. Select the item to be deleted.
2. Do one of the following:
   - Press `Alt+Delete`.
   - Select Refactor | Safe Delete from the main or the context menu.
   - Select Refactor | Refactor This from the main menu (`Ctrl+Shift+Alt+T`), and select Safe Delete.
3. In the Safe Delete dialog , select the necessary options and click OK.
4. If the refactoring is potentially unsafe, the Usages Detected dialog opens.
   - View Usages. Click this button to see where in your code the item you are about to delete is used. As a result, the Find tool window opens.
     Analyze you code and make the necessary corrections. Then click Do Refactor. (If you want to rerun the refactoring from its start, click Rerun Safe Delete. PhpStorm will check if the refactoring is safe once more.)
   - Delete Anyway. Click this button to delete the item without looking at its usages.

Analyzing Applications

In this part:

- Viewing Structure and Hierarchy of the Source Code
- Analyzing Data Flow
- Analyzing External Stacktraces
- Navigating from Stacktrace to Source Code
- Viewing PSI Structure
- Analyzing Duplicates

Viewing Structure and Hierarchy of the Source Code

PhpStorm enables you to examine the hierarchy of classes, methods, and calls in the Hierarchy tool window, and explore the structure of source files in the Structure tool window.

- Both the Hierarchy and the Structure tool windows are available from the View menu.
- the Hierarchy tool window only becomes available, when a hierarchy is built.
- Hierarchies are built in the Navigate menu.

In this part:

- Building Hierarchies
- Viewing Hierarchies
- Viewing Structure of a Source File

Building Hierarchies

PhpStorm provides facilities for building the following hierarchies:

- Type hierarchies that show parent and children classes of a class.
- Method hierarchies that show classes where a method:
  - Is defined.
  - Is not defined.
  - Should be defined if the class is not abstract.
- Call hierarchies that show callers (supertypes) or callees (subtypes) of a method.

When built, a hierarchy can be immediately viewed and examined in the Hierarchy tool window. By default, every new built hierarchy overwrites the contents of the current tab. You can retain the current tab and have the next hierarchy built in a new one.

To build a hierarchy of types

1. Select the desired class in the Project tool window or open it in the editor.
2. On the main menu, choose Navigate | Type Hierarchy or just press `Ctrl+H`.

To build a method hierarchy

1. Open the file in the editor and place the caret at the declaration of the desired method. Alternatively, select the desired method in the Project view.
2. On the main menu, choose Navigate | Method Hierarchy or press `Ctrl+Shift+H`.

To build a hierarchy of method calls

1. Open the file in the editor and place the caret at the declaration or usage of the desired method. Alternatively, select the desired method in the Project view.
2. On the main menu, choose Navigate | Call Hierarchy or press `Ctrl+Alt+H`.

To retain a hierarchy tab

- In the Hierarchy tool window, click the Pin Tab button on the toolbar.

Viewing Hierarchies

Once built , hierarchies can be brought up for close examination in the Hierarchy tool window .

On this page:

- Showing the Hierarchy tool window
- Navigating between the tabs of the Hierarchy tool window
- Toggling between views

To show the Hierarchy tool window, do one of following

Warning! The Hierarchy tool window is not shown when there are no hierarchies to display. You have to build hierarchies first.

Refer to the section Building Hierarchies to learn how to build hierarchies.

- On the main menu, choose View | Tool Windows | Hierarchy.
- Use `Alt+8` keyboard shortcut.

To navigate between the tabs of the Hierarchy window, do one of the following

- Right-click the currently displayed tab, and choose Select Next Tab/ Select Previous Tab on the context menu.
- Use the `Alt+Right` and `Alt+Left` keyboard shortcuts.
- Click the currently displayed tab, and choose the next tab to display.

To toggle between views, use the toolbar of the Hierarchy tool window

Toggling between views means showing ascending or descending hierarchy (callee vs. caller methods, parent vs. children classes etc.)

- Click to show caller methods.
- Click to show callee methods.

Viewing Structure of a Source File

On this page:

- Basics
- Viewing the structure of a file
- Viewing members

Basics

You can examine the structure of the file currently opened in the editor using the Structure tool window or the Structure pop-up window.

By default, PhpStorm shows all the classes, methods, etc. presented in the current file.

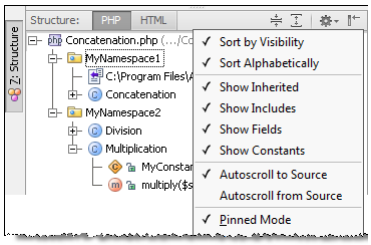
To have other members displayed , click the corresponding buttons on the toolbar of the Structure tool window.

You can also have class members shown in the Project tool window.

Viewing the structure of a file

To view the file structure, do one of the following

- On the main menu, choose View | Tool Windows | Structure to show the Structure tool window .
- Press Structure tool button to show the Structure tool window .
- Press `Alt+7` to show the Structure tool window .
- Press `Ctrl+F12` to show the Structure popup .



Viewing members

To have class fields displayed

- Click on the toolbar of the Structure tool window.

To have constants displayed

- Click on the toolbar of the Structure tool window.

To have inherited members displayed

- Click on the toolbar of the Structure tool window.

By default, PhpStorm shows only methods, constants, and fields defined in the current class. If shown, inherited members are displayed gray.

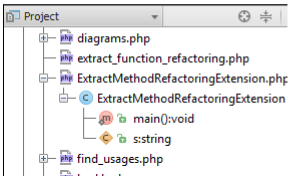
To have included files displayed

- Click on the toolbar.

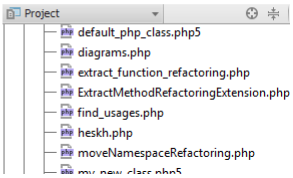
To have class members shown in the Project tool window

- Turn on the Show members item on the context menu of the Project tool window title bar. If this option is on, the files in the tree that contain classes turn into nodes. When such node is unfolded, the contained classes with their fields, methods, and other members of the selected item are shown.

Show Members on



Show Members off



Analyzing Data Flow

On this page:

- Introduction
- Analyzing data flow
- Examining the results of Dataflow analysis
  - Dataflow to Here
  - Dataflow from Here

Introduction

PhpStorm provides Dataflow Analysis features to help you with code archeology - better understand the inherited project code, interpret complicated parts of the code, find bottlenecks in the source, and more.

Specifically, Dataflow to/from here features allow you to:

- See where values assigned to a variable come from.
- Find out all the possible values that a variable can have.
- Find out the places an expression\variable\method parameter can flow into.

If you want to trace back the value passed to a parameter at caret, you can create a slice view of the source code using the Analyze | Dataflow to Here command. Moreover, using Analyze | Dataflow from Here command you can find the places an expression can flow into. The results of each dataflow analysis are displayed in the dedicated tabs of the Analyze Dataflow tool window.

Analyzing data flow

To analyze dataflow to/from a symbol:

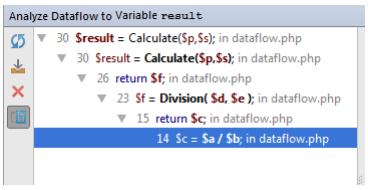
1. Open the desired file for editing, see Opening and Reopening Files in the Editor .
2. Place the caret at the symbol to be analyzed (expression\variable\method parameter).
3. On the main menu, or on the context menu, choose Code | Analyze Dataflow to Here or Code | Analyze Dataflow from Here depending on your purposes.
4. Specify the analysis scope and choose whether you want to ignore all values coming from the test code.
5. Click OK. Review the analysis results in the dedicated Analyze Dataflow tool window.

Examining the results of Dataflow analysis

The following sections briefly describe how to "read" the dataflow analysis results.

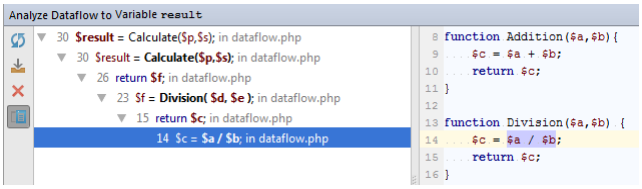
Dataflow to Here

- Expand the tree to dig in the chain of assignments and method calls that lead to the symbol in question. Nodes with gray background denote duplicates (the usages that are already present in the tree in another location). The image below shows an example of the Dataflow to Here analysis results:



  The values in this view flow in the following direction: the value for the field `$result` in `dataflow.php` comes from the `$result = Calculate($p,$s)` assignment statement where the `Calculate($m,$n)` method called with the `$p` and `$s` parameters returns `$f` . The value of `$f` comes from the `Division($a,$b)` method called with the `$d` and `$e` parameters and so on.

- To view the code of assignments and method calls, press the toggle button. PhpStorm adds a Preview pane which shows the code of the assignment or call that is currently selected in the tree, the code is highlighted:



- To navigate to the source code of an assignment or call, double click the relevant line in the tree.

Dataflow from Here

The hierarchical view is similar to the results of Dataflow to Here analysis, but the values flow in the opposite direction.

Analyzing External Stacktraces

On this page:

– Overview

– Analyzing external stacktrace

Overview

You might want to analyze exceptions received by someone else, for example, QA engineers, or investigate a deadlock, or a hang-problem. Unlike the exceptions that you get in the debug mode or when running unit tests, these exceptions do not have links that help you navigate to the corresponding locations in the source code.

With PhpStorm, you can simply copy an exception or full thread dump, paste it to the Stacktrace Analyzer, explore information, and navigate to the corresponding source code.

Analyzing external stacktrace

To analyze an external stack trace or thread dump

1. On the main menu, choose Tools | Analyze Stacktrace.
2. In the Analyze Stacktrace dialog box that opens, paste the external stack trace or thread dump into the Put a thread dump here: text area.
3. Click OK. The stacktrace is displayed in the Run tool window.

Navigating from Stacktrace to Source Code

You can easily navigate from the stack trace in the Run tool window to the source code that causes problems.

To navigate from the stack trace to a line of code

- In the Run tool window scroll to the desired stack trace line and click the link to the source file in question. The source file opens in the editor.

Viewing PSI Structure

If you want to explore the internal structure of source code as it is interpreted by PhpStorm, use the PSI viewer.

Important note

PSI viewer command is only available when there is at least one plugin module in project.

If you want this information to be available for any project, add the following line to the file `bin/idea.properties` under the product installation:

```
idea.is.internal=true
```

To view PSI structure of the source code

1. On the main menu, choose Tools | View PSI Structure.
2. In the PSI Viewer dialog box, type or paste the fragment of source code to be analyzed in the Text area, select file type, and specify the other options.
3. Click Build PSI Tree, and preview the generated PSI tree in the PSI Structure pane.

   If the source code in the Text area has been changed, click the same button to refresh preview.

Analyzing Duplicates

On this page:

– Overview

– Searching for duplicates

Overview

PhpStorm helps you find repetitive blocks of code in a certain range. This range can be a single file, a project, a module, or a custom scope. Results of analysis display in the dedicated tab of the Duplicates tool window .

Searching for duplicates

To search for duplicates

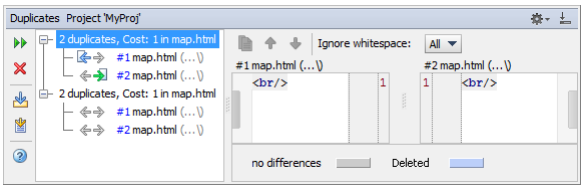
1. On the main menu, choose Code | Locate Duplicates.

2. In the Specify Code Duplication Analysis Scope dialog , specify the analysis scope (whole project, current file, uncommitted files (for the projects under version control), or some custom scope). In addition, you can include test sources into the analysis too.
   Click OK, when ready.

3. In the Code Duplication Analysis Settings dialog , do the following:
   1. Select languages to perform analysis in.
   2. For each language, check the options to define your preferences for the analysis.
      For example, you can opt to request identical match for code fragments to be considered duplicates, or specify a certain limit below which the code constructs are not considered duplicates (to avoid reporting about each `if` construct in the source code).

   Click OK.

4. In the Duplicates tool window , explore search results.



   – View the list of duplicates in the left pane of the tool window.
   – View differences between the found duplicates in the right pane. Use the arrow buttons to place the selected duplicate in one of the sections of the differences viewer and compare fragments of the code.
   – Navigate to the duplicates in the editor, using Jump to Source or Show Source commands of the duplicates context menu.
   – Eliminate duplicates from the source code by applying the Extract method refactoring to the detected repetitive blocks of code that are found and highlighted automatically.

Working with Run/Debug Configurations

In this section:

- Creating and Editing Run/Debug Configurations
- Changing Run/Debug Configuration Defaults
- Creating and Saving Temporary Run/Debug Configurations
- Creating Folders and Grouping Run/Debug Configurations

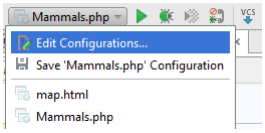
Creating and Editing Run/Debug Configurations

On this page:

- Overview
- Creating a run/debug configuration
- Editing an existing run/debug configuration

Overview

With the Navigation bar visible (View | Navigation Bar), the available run/debug configurations are displayed in the run/debug configuration selector in the Run area:



PhpStorm provides the Run/Debug Configuration dialog box as a tool for handling run/debug configurations: create configuration profiles or change the default ones.

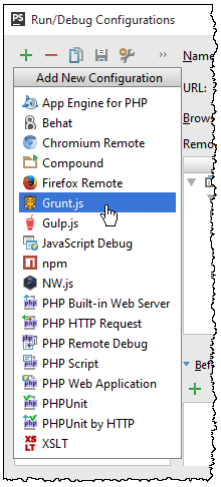
PhpStorm suggests the following ways to create a run/debug configuration:

- Create a run configuration manually on the base of the default one, using the Run/Debug Configuration dialog box.
- Save a temporary run configuration .

Creating a run/debug configuration

To create a run/debug configuration, follow these steps:

1. In the Run/Debug Configuration dialog box, click + on the toolbar or press `Alt+Insert`. The drop-down list shows the default run/debug configurations. Select the desired configuration type.



   The fields that appear in the right pane, display the default settings for the selected configuration type.

   Note If you want to change the settings of the default run/debug configuration, expand the Defaults node, select the desired configuration type, and modify it as required.

2. For the new run/debug configuration:
   - Specify its name in the Name text box. This name will be shown in the list of the available run/debug configurations.
   - Specify whether you want to make PhpStorm check execution status of the instances of the same run/debug configuration. If you want to make sure that only one instance of the run/debug configuration is currently executed, select the check box Single instance only. In this case, a confirmation dialog box will show up every time you try to launch run/debug configuration, when one instance of the same type is still running.
     If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place.

     If this check box is not selected, you can launch as many instances of the runner as required. As the result, each runner will start in its own tab of the Run tool window.

   Tip To use an existing configuration as a pattern, create its copy by clicking the Copy button on the toolbar, then change it as required.

3. In the Before launch section, define whether you want to execute some tools or scripts prior to launching the run/debug configuration.
4. Specify additional parameters depending on the configuration type. Refer to the descriptions of run/debug configuration parameters below the Run/Debug Configurations section.
5. Apply the changes and close the dialog box.

Editing an existing run/debug configuration

To change an existing run/debug configuration:

- On the main menu, choose Run | Edit Configurations.
- With the Navigation Bar visible (View | Navigation Bar), choose Edit Configurations from the run/debug configurations selector.
- Press `Shift+Alt+F10`, then press `0` to display the Edit Configuration dialog box or select the configuration from the pop-up window and press `F4`.

In the corresponding run/debug configuration dialog box, change parameters as required.

Changing Run/Debug Configuration Defaults

On this page:

– Introduction

– Configuring defaults

Introduction

For a run/debug configuration of a particular type, you can set up the default values for one or more parameters and save them as template for further usages. In this case, next time when you create a new configuration of that type, the corresponding fields of the dialog will already contain the specified values.

Note that changing the default values does not affect the already existing run/debug configurations.

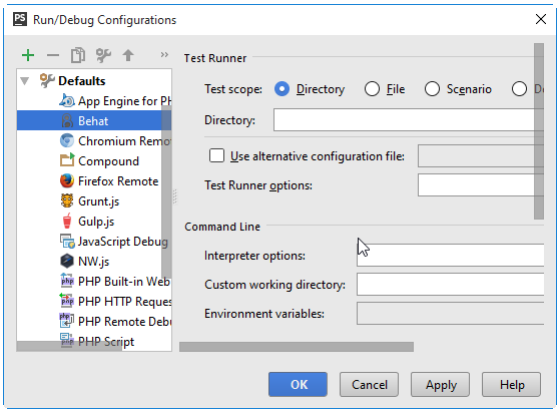
Configuring defaults

To set up the default values for a run/debug configuration, follow these steps:

1. In the left-hand pane of the run/debug configuration dialog, expand the Defaults node.

2. From the drop down list, select the desired configuration type. The corresponding configuration template appears in the right-hand pane.

3. Specify the desired parameters and click Apply to save the template.



Tip The fields Name, Share and Single instance only are not available for the default run/debug configurations.

Creating and Saving Temporary Run/Debug Configurations

On this page:

- Introduction
- Creating a temporary run/debug configuration
- Saving a temporary run/debug configuration

Introduction

Sometimes you might need to run or debug a certain class or file , without creating a dedicated run configuration. In this case, PhpStorm provides a temporary run configuration .

Temporary run/debug configuration is added to the list of available configurations and works same way as the permanent run/debug configurations . You can change its settings using the Run/Debug Configuration dialog box and optionally save it as permanent.

Creating a run/debug configuration involves specifying the URL address of the Web page to open. For temporary configurations, PhpStorm assembles this URL address automatically. To have it done properly, you need to specify the project root folder URL on the PHP page of the Settings dialog box.

Creating a temporary run/debug configuration

To create a temporary run/debug configuration

1. Select the desired PHP class or file in the Project tool window or open it in the editor.
2. Do one pf the following:
   - On the context menu, choose Run <name> or Debug <name>.
   - Press `Ctrl+Shift+F10`.

   PhpStorm creates a temporary configuration, which appears in the Run/Debug Configuration selector, when the run or debug session is over.

Saving a temporary run/debug configuration

To save a temporary configuration, do one of the following

- In the Run/Debug Configuration selector, choose Save <configuration name>.
- In the Run/Debug Configuration dialog box, click .
- On the context menu of the editor or Project view, choose Save <configuration name>.

Creating Folders and Grouping Run/Debug Configurations

On this page:

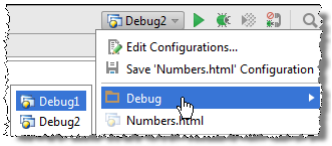
- Introduction
- Creating folders of run/debug configurations
- Deleting folders
- Changing order of folders

Introduction

When there are too many run/debug configurations of the same type, you can put them into directories, and thus make them visually distinguishable.

Folders area used to organize run/debug configurations. When not needed, they can be deleted, and the run/debug configurations under those folders just pass under the root of the corresponding type.

Once grouped, the run/debug configurations appear in the drop-down list under the corresponding folders:



Creating folders of run/debug configurations

To create a folder of run/debug configurations

1. Open Run/Debug Configuration dialog box.
2. In the Run/Debug Configuration dialog box , click on the toolbar. A new empty folder is created.
3. Specify the folder name in the text field to the right, or accept default.
4. Select the desired run/debug configuration of a certain type, and move under the target folder. This can be done in one of the following ways:
   - Use drag-and-drop.
   - Use the toolbar buttons .
   - Press `Alt+Up` or `Alt+Down`.
5. Apply changes. Note that the folders of run/debug configurations should not be empty. The empty folders are not saved.

Deleting folders

To delete a folder

1. In the Run/Debug Configuration dialog box , select a folder to be deleted.
2. On the toolbar, click . The selected folder is deleted silently. Any run/debug configurations grouped under this folder, are moved under the root of the corresponding type.
3. Apply changes.

Changing order of folders

To move folders up or down

1. In the Run/Debug Configuration dialog box , select one of the folders within a category of run/debug configurations.
2. Do one of the following:
   - On the toolbar, click or .
   - Press `Alt+Up` or `Alt+Down`.

   The folder in question moves one position up or down.
3. Apply changes.

After closing the dialog box, groups of run/debug configurations in the run/debug configurations selector on the main toolbar appear in the order achieved by moving folders up or down within a category.

Running

This section describes the procedures that are common for the various types of applications:

- Running Applications
- Rerunning Applications
- Reviewing Results
- Stopping and Pausing Applications
- Viewing Running Processes

For the details related to running applications in the supported frameworks, refer to Language and Framework - Specific Guidelines

Running Applications

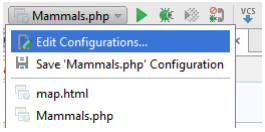
On this page:

- Introduction
- Running an application

Introduction

PhpStorm enables running entire applications as well as particular classes or files .

PhpStorm makes use of the settings defined in a Run/Debug Configuration . All the run configurations that exist in a project, are available in the Select Run/Debug Configuration drop-down list.



If you want to see a list of all currently running applications, select Run | Show Running List from the main menu. Refer to the section Viewing Running Processes for details.

Running an application

To run an application

- Do one of the following:
  - In the left gutter, click the icon ▶, and choose the desired command.
  - On the main toolbar, select the desired run configuration, and:
    - Choose Run | Run from the main menu.
    - Click ▶.
    - Press `Shift+F10` .
  - Press `Shift+Alt+F10` , select the desired run configuration from the pop-up menu, and press `Enter` .

From this pop-up menu you can:

- Invoke the Edit Configuration dialog.
- Edit the selected configuration before launch (`F4`).
- Instantly delete a temporary configuration (`Delete`).
- Switch from run to debug and vice versa (hold `Shift`).
- Access a previously selected configuration (`1`).
- Access context-dependent configuration (`2` or `3`).

This pop-up menu can also be quickly accessed by pressing `F9` , when you're not running any debug session.

Note If the options that launch tools before running were enabled, PhpStorm runs the tools , and after success will run the application. Otherwise, the program will start immediately.

Note that after you've started a run session, the ▶ icon that marks the Run tool window and in the Run/Debug Configuration Selector toggles to ▶ to indicate that the run process is active.

Rerunning Applications

You can re-run an application if its tab is still opened in the Run window. The program re-runs with the initial settings.

To re-run an application

1. In the Run window, select the tab where the desired application is opened.
2. In the toolbar of the Run window, click the Rerun button , or press `Ctrl+F5`.

   Tip If you want to re-run without loosing focus in the editor tab you are working in, press `Shift+F10`.

Reviewing Results

You can review any output from your running applications in the Run window console. The output from each application is displayed in its own tab of the Run tool window, named after the corresponding run/debug configuration .

If you re-run an application, the new output overwrites the contents of the tab. To preserve the output of an application, even if you re-run it, pin the output tab.

Stopping and Pausing Applications

In the Run tool window, you can stop a program, or pause its output. If a program is stopped, its process is interrupted and exits immediately. When program output is paused, the program continues running in the background, but its output is suspended.

To stop a program, do one of the following

- In the Run tool window, click the Stop button on the toolbar, or press `Ctrl+F2`.
- To close the active tab, click the Close button, or press `Ctrl+Shift+F4`.

To suspend program output

- In the Run tool window, click the Pause button on the toolbar.
  Note that the button is not available for Run/Debug Configuration: Node JS , Run/Debug Configuration: Node JS Remote Debug , and Run/Debug Configuration: NodeUnit .

Viewing Running Processes

PhpStorm makes it possible to view all the running applications. The command Show Running List of the menu Run is only enabled if there are active applications. If no applications are active, the command is greyed out.

To view the list of running applications

- On the main menu, choose Run | Show Running List.

A popup listing all active applications is displayed in the top-right corner of the editor.

Tip Same is also valid for debugging.

Debugging

- Debugging
  - Overview
  - General debugging steps
- Using Breakpoints
- Configuring Debugger Options
- Starting the Debugger Session
- Pausing and Resuming the Debugger Session
- Examining Suspended Program
- Stepping Through the Program
- Finding the Current Execution Point
- Monitoring the Debug Information
- Live Editing of HTML, CSS, and JavaScript
- Viewing Actual HTML DOM
- Using JetBrains Chrome Extension
- Inline Debugging

Overview

This section describes the procedures that are common for various types of applications.

For details on debugging applications in the supported frameworks, refer to Language and Framework - Specific Guidelines .

PhpStorm provides a full range of facilities for debugging your source code:

- Breakpoints in PHP.
- Breakpoints in HTML, and JavaScript.
- Customizable breakpoint properties: conditions, pass count, etc.
- Frames , variables , and watches views in the debugger UI.
- Runtime evaluation of expressions .
- Support for the Xdebug tool to debug PHP applications .

If you want to see a list of all currently debugging applications, select Run | Show Running List from the main menu. Refer to the section Viewing Running Processes for details.

General debugging steps

1. Configure the debugger options .
2. To debug CoffeeScript, TypeScript, and Dart code, you need source maps generated in addition to the JavaScript code.
   Source maps set the correspondence between lines in your original code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.

   JavaScript and source maps are generated by transpiling the original code manually using the File Watcher of the corresponding type (CoffeeScript, TypeScript, or Dart). After that, you can debug the output JavaScript code.

   For details, see Using File Watchers , Transpiling CoffeeScript to JavaScript , and Transpiling TypeScript to JavaScript .
3. Define a run/debug configuration for the application to be debugged.
4. Create breakpoints in the source code.
5. Launch a debugging session.
6. Pause or resume the debugging session as required.
7. During the debugger session, step through the breakpoints , evaluate expressions , change values on-the-fly , examine suspended program , and set watches .

After you've started a debug session, the icon that marks the Debug tool window toggles to to indicate that the debug process is active.

Using Breakpoints

In this section:

- Accessing Breakpoint Properties
- Configuring Breakpoints
- Creating Line Breakpoints
- Creating Exception Breakpoints
- Enabling, Disabling and Removing Breakpoints
- Moving Breakpoints
- Navigating Back to Source
- Named Breakpoints
- Working with Groups of Breakpoints

Accessing Breakpoint Properties

On this page:

- Introduction
- Viewing all breakpoints
- Viewing properties of a breakpoint

Introduction

To view the whole list of the breakpoints in the current project, use the Breakpoints dialog box. For each individual breakpoint in the list, you can view and change its properties as required.

Viewing all breakpoints

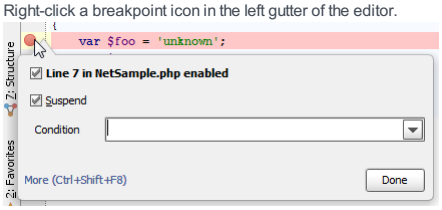
To view the list of all breakpoints and their properties, do one of the following:

- On the main menu, choose Run | View Breakpoints.
- Press `Ctrl+Shift+F8`.
- In the toolbar of the Debug tool window , click .
- Breakpoints are visible in the Favorites tool window.

Viewing properties of a breakpoint

To view properties of a single breakpoint

- Right-click a breakpoint icon in the left gutter of the editor.



Configuring Breakpoints

On this page:

- Basics
- Configuring breakpoints

Basics

For a breakpoint, you can configure the following properties:

- Actions to be performed upon hitting a certain breakpoint.
- Suspend policy, which defines whether the application should be suspended upon hitting the breakpoint.
- Dependencies on other breakpoints.
- Conditions defining when a breakpoint is hit.

PhpStorm suggests the following way to change the breakpoints properties:

- Using the Breakpoints dialog box, for a breakpoint selected in the list.
- Using breakpoint icon in the left gutter

Configuring breakpoints

To configure actions, suspend policy and dependencies of a breakpoint

1. Do one of the following:
   - Right-click a breakpoint in the left gutter, and then click the link More or press `Ctrl+Shift+F8`.
   - Open the Breakpoints dialog box as described on page Accessing Breakpoint Properties and select the desired breakpoint in the list.
   - In the Favorites tool window, select the desired breakpoint, and click .

   Note that the pop-up window shows less options than the Breakpoints dialog box. To show hidden options, click More.

2. Define the actions to be performed by PhpStorm on hitting breakpoint:
   - To notify about the reaching of a breakpoint with a text message in the debugging console, select the Log message to console check check box.
     To evaluate an expression in the context of a breakpoint and display its value in the debugging console, check the option Evaluate and log, and enter a valid expression in the option field.

     This feature lets you obtain information about your running application without having to suspend its execution.
   - To set a breakpoint the current one depends on, select it from the Disabled until selected breakpoint hit drop-down list. Once dependency has been set, the current breakpoint is disabled until selected one is hit.
     - Choose Disable again radio button to disable the current breakpoint after selected breakpoint was hit.
     - Choose Leave enable radio button to keep the current breakpoint enabled after selected breakpoint was hit.
   - Enable suspending an application upon reaching a breakpoint by selecting the Suspend check box, and then select one of the option buttons to specify the way a running program will be paused. For more information on the Suspend options, refer to Breakpoints dialog reference.

Creating Line Breakpoints

On this page:

- Basics
- Creating line breakpoints in the editor
- Creating temporary line breakpoints
- Deleting line breakpoints

Basics

A line breakpoint is a breakpoint assigned to a specific line in the source code.

Line breakpoints can be set on executable lines. Comments, declarations and empty lines are not valid locations for the line breakpoints.

Creating line breakpoints in the editor

1. Place the caret on the desired line of the source code.
2. Do one of the following:
   - Click the left gutter area at a line where you want to toggle a breakpoint.
   - On the main menu, choose Run | Toggle Line Breakpoint.
   - Press `Ctrl+F8`.

Creating temporary line breakpoints

1. Place the caret on the desired line of the source code.
2. Do one of the following:
   - On the main menu, choose Run | Toggle Temporary Line Breakpoint.
   - Press `Ctrl+Shift+Alt+F8`.

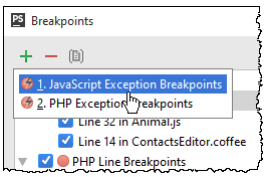
Deleting line breakpoints

Do one of he following:

- In the Breakpoints dialog box, select the desired line breakpoint, and click .
- In the editor, locate the line with the line breakpoint to be deleted, and click its icon in the left gutter.
- Place caret on the desired line and press `Ctrl+F8`.

Creating Exception Breakpoints

1. On the main menu, choose Run | View Breakpoints, or press `Ctrl+Shift+F8`.
2. In the Breakpoints dialog box that opens, click +.
3. Select PHP Exception Breakpoint or JavaScript Exception Breakpoint from the drop-down list.



4. In the Add Exception Breakpoint dialog box that opens, specify the custom Exception type to break on or use a standard Warning, Notice, or Deprecated to break on PHP error conditions. Note that E_ERROR, E_PARSE, and E_COMPILE_ERROR are not handled as they halt execution of the PHP engine. Click OK when ready. PhpStorm returns you to the Breakpoints dialog box.
5. Configure the new exception breakpoint as described in Configuring Breakpoints .

Enabling, Disabling and Removing Breakpoints

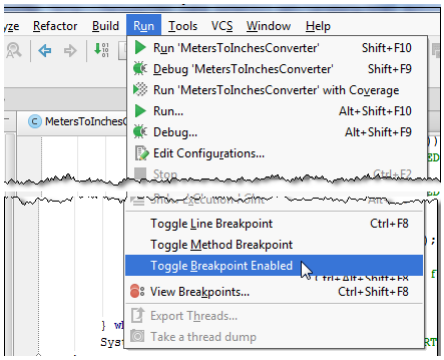
On this page:

- Toggling between the enabled and disabled state of a breakpoint
- Disabling a breakpoint temporarily in the editor

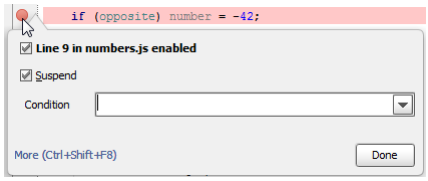
Toggling between the enabled and disabled state of a breakpoint

When you temporarily disable or enable a breakpoint, its icon changes from to and vice versa.

1. Place the caret at the desired line with a breakpoint.
2. Do one of the following:
   - On the main menu, choose Run | Toggle Breakpoint Enabled:



   - Right-click the desired breakpoint icon, select or deselect the <breakpoint name> enabled check box, and then click Done.



   - `Alt` -click on the breakpoint icon.

Tip Alternatively, open the Breakpoints dialog box, select the desired breakpoint, and select or clear the check box to its left.

Disabling a breakpoint temporarily in the editor

When you temporarily disable a breakpoint, its icon changes from to .

Alternatively, open the Breakpoints dialog box, as described on page Accessing Breakpoint Properties , select the desired breakpoint, and clear the check box next to it or the Line <line number> in <file name> check box in the right-hand pane.

Enabling a temporarily disabled breakpoint in the editor

When you enable a temporarily disabled breakpoint, its icon changes from to .

1. Place the caret at the desired line with a breakpoint.
2. Do one of the following:
   - Right-click the desired breakpoint icon, select the Line <line number> in <file name> check box in the pop-up dialog box that opens, and then click Done.
   - With the `Alt` key pressed, click the breakpoint icon.

Alternatively, open the Breakpoints dialog box, as described on page Accessing Breakpoint Properties , select the desired breakpoint, and select the check box next to it or the Line <line number> in <file name> check box in the right-hand pane.

Removing a breakpoint

To remove a breakpoint permanently, do one of the following:

- Open the Breakpoints dialog box , select the desired breakpoint, and click .
- Click the breakpoint icon in the left gutter of the editor.

Removing all breakpoints of a certain type

1. On the main menu, choose Run | View Breakpoints, or press `Ctrl+Shift+F8`.
2. In the Breakpoints dialog, press the left arrow key to select the desired category.
3. Press `Delete`.

All breakpoints of a certain type will be deleted.

Moving Breakpoints

To move a breakpoint, drag-and-drop it to the target line.

To navigate to the breakpoints source from the Breakpoints dialog box

- To view the selected breakpoint without closing the dialog box, use the preview pane.
- To open the file with the selected breakpoint for editing, double-click the desired breakpoint. To close Breakpoints dialog, press `F4`. The caret will be placed at the line marked with the breakpoint in question.

Named Breakpoints

On this page:

- Introduction
- Editing breakpoint description
- Searching for a breakpoint using its name

Introduction

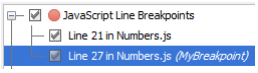
PhpStorm makes it possible to add a name or a short description to a breakpoint to facilitate search.

Editing breakpoint description

To edit a breakpoint description

1. Open the Breakpoints dialog .
2. Right-click a breakpoint you are interested in.
3. On the context menu, choose Edit description.
4. In the Edit Description dialog box, type the desired description.

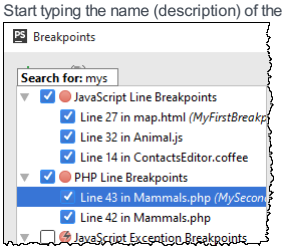
The specified description shows in italic next to the address of a breakpoint in the Breakpoints dialog:



Searching for a breakpoint using its name

To search a brekpoint by name

1. Open the Breakpoints dialog .
2. Start typing the name (description) of the desired breakpoint.



The breakpoint with the matching description gets the focus.

Working with Groups of Breakpoints

On this page:

- Introduction
- Creating groups of breakpoints
- Moving breakpoints to another group, or out of a group
- Toggling a group of breakpoints

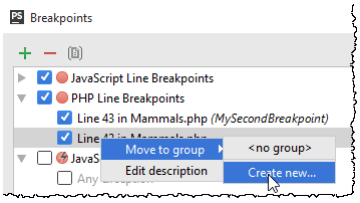
Introduction

PhpStorm makes it possible to organize breakpoints in groups, for example, to mark out breakpoints for a specific problem. This is done in the Breakpoints dialog .

Creating groups of breakpoints

To create a group of breakpoints

1. In the Breakpoints dialog , right-click one or more breakpoints you are interested in.
2. On the context menu, point to the command Move to group, and then on the submenu, choose Create new...:

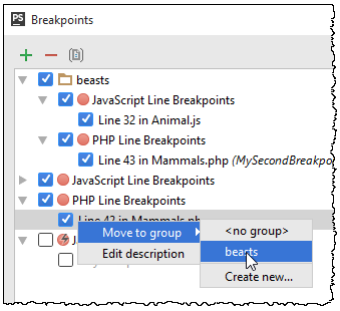


3. In the New Group dialog box, type the name of the new group. The selected breakpoint moves to the newly created group.
4. Optionally, you can right-click a group of breakpoints and select Set as default from the popup menu. All newly created breakpoints will be automatically added to this group.

Moving breakpoints to another group, or out of a group

To move breakpoints to another group

1. In the Breakpoints dialog , right-click one or more breakpoints you are interested in.
2. On the context menu, point to the command Move to group, and then on the submenu, choose the desired group name:



The breakpoints in question move to the selected group.

To move breakpoints out of a group

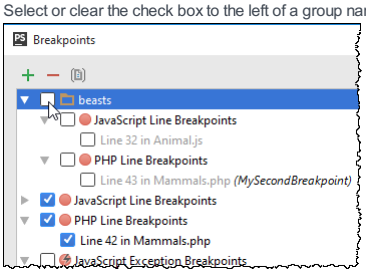
1. In the Breakpoints dialog , right-click one or more breakpoints within a group.
2. On the context menu, point to the command Move to group, and then on the submenu, choose <no group>.
3. The selected breakpoints move to a node according to their type .

Toggling a group of breakpoints

Using groups of breakpoints, it is possible to toggle all breakpoints within a group in a single click.

To toggle all breakpoints belonging to a certain group

- Select or clear the check box to the left of a group name:



Configuring Debugger Options

On this page:

– Introduction

Introduction

PhpStorm supports debugging for PHP applications, classes, and files. The debugging functionality is incorporated in PhpStorm, you only need to configure its settings.

Depending on the plugins enabled, PhpStorm can also support debugging for other languages, for example, JavaScript .

The JavaScript debugging functionality is incorporated in PhpStorm, you only need to configure its settings .

PhpStorm supports local and remote JavaScript debugging only in Google Chrome and other browsers of the Chrome family.

To debug PHP applications, you need to configure a dedicated tool .

Starting the Debugger Session

On this page:

- Before debugging
- Debugging an application

Before debugging

- Set breakpoints in the source code.
- If necessary, create or modify the corresponding Run/Debug configuration .

The debug session starts with the selected run/debug configuration. Note that several debug processes can be launched simultaneously.

Debugging an application

To start debugging an application, do one of the following

- Select the run/debug configuration to execute, and then do one of the following:
  - Click icon in the left gutter, and then choose .
  - Click on the toolbar.
  - Choose Run | Debug on the main menu.
  - Press `Shift+F9`.
- Press `Shift+Alt+F9`, select the configuration from the pop-up menu, and press `Enter`.

Note that after you've launched a debug session, the icon that marks the Debug Tool Window toggles to to indicate that the debugging process is active.

Pausing and Resuming the Debugger Session

On this page:

- Introduction
- Pausing the debugger session
- Resuming the debugger session

Introduction

When a breakpoint is hit, or when a running thread or an application is paused manually, the debugging session is suspended.

Pausing the debugger session

To pause the debugger session, do one of the following

- On the main menu, choose Run | Pause Program.
- Click on the Debug toolbar.
  Note that the button is not available for Run/Debug Configuration: Node JS , Run/Debug Configuration: Node JS Remote Debug , and Run/Debug Configuration: NodeUnit .

Resuming the debugger session

To resume the debugger session, do one of the following

- On the main menu, choose Run | Resume Program.
- Click on the Debug toolbar.
- Press F9 .

Examining Suspended Program

On this page:

– Basics

– Examining a suspended thread

– Navigating between frames

Basics

When a breakpoint is hit, or a program execution is manually suspended , you can examine your application by analyzing frames.

A frame corresponds to an active method or function call. A frame stores the local variables of the called method or function, the arguments to it, and the code context that enables expression evaluation.

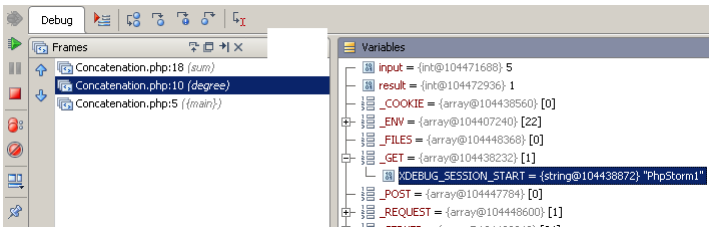
All currently active frames are displayed on the Frames pane of the Debug tool window, where you can switch between them and analyze the information stored therein.

Examining a suspended thread

To examine frames of a suspended thread

1. Select a thread from the thread selector drop-down list on top of the Frames pane. The list of frames is displayed:

2. Select a frame from the Frames list. The Variables pane shows all the variables available to the method call in this frame, so you can further explore them.



Navigating between frames

Do one of the following:

– Use up and down arrow buttons on the toolbar.

– Use `Up` and `Down` shortcuts.

You do not need to perform any actions to navigate to the frame's source code. PhpStorm automatically jumps to the source code of the selected frame in the editor.

Adding, Editing and Removing Watches

On this page:

- Introduction
- Creating watches
- Editing watches
- Deleting watches

Introduction

If you want to evaluate a number of variables or expressions in the context of the current frame, and view all of them simultaneously, you can create watches for them. The values of the expressions are updated with each step through the application, but are only visible when the application is suspended. Unlike the Expression Evaluation feature , these expressions are persisted as the part of your project.

This section describes how to add items to watches, change and remove watches.

Creating watches

To add an item to a watch

Do one of the following:

- In the Watches pane, click +, or just press `Insert`.
- On the Variables pane, in the Inspection window, or in the Evaluate Expression dialog box, right-click the desired item and choose Add to Watches on the context menu.
- Select the desired item in the Variables pane and drag it to the Watches pane.
- Select item in the editor, right-click it and select Add to Watches on the context menu.

Tip You can navigate from a backtrace in the Watches pane to the respective line of the source code. To do that, right-click a line of backtrace, and choose Jump to Source on the context menu, or just press `F4`.

Editing watches

To edit a watch

- To change the expression represented by a watch, right-click the desired watch and select Edit on the context menu.

Deleting watches

To remove a watch

1. In the Watches pane, select a watch to be deleted.
2. On the context menu, choose Remove Watch, or press `Alt+Delete`.

Inspecting Watched Items

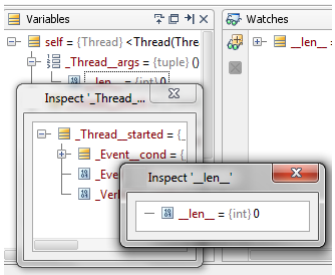
On this page:

- Introduction
- Inspecting references

Introduction

PhpStorm helps inspect any variables or watches item in its own window. For example, if you need to examine several references in detail, you can open an inspection window for each of them. So doing, a separate window is created for each variable or watch reference and all of its child references.

The inspection windows are non-modal, and you can launch as many as of them required. All changes of the references are immediately reflected in the corresponding inspection windows.



Inspecting references

To inspect a reference

1. Select the item to be inspected on the Variables or Watches pane.
2. On the context menu, choose Inspect.

Evaluating Expressions

On this page:

- Basics
- Limitations
- Evaluating arbitrary expressions
- Evaluating expressions in the editor
- Evaluating variables in the console pane
- Executing code in the console pane during a debugging session

Basics

PhpStorm enables you to evaluate arbitrary single-line expressions and provides a way to quickly evaluate an expression in the editor at caret or a selection.

Limitations

While using the Expression Evaluation feature, be aware of the following:

- A method can be invoked within the Expression Evaluation dialog only if the debugger has stopped at a breakpoint, but has not been paused.
- Expression Evaluation can only be "single-level". In other words, if PhpStorm stops at a breakpoint within a method called from the Expression Evaluation, you cannot use the Expression Evaluation feature again.

> **Tip** Note that in certain operating systems the key and mouse combinations may not work as described here. In this case, it's necessary to tweak the operating system's keymap. For example, if you are using Ubuntu, mind the windows manager, whose shortcuts conflict with that of PhpStorm.

Evaluating arbitrary expressions

1. Open the Evaluate Expression dialog box in one of the following ways:
   - Choose Run | Evaluate Expression on the main menu.
   - Press `Alt+F8`.
   - To evaluate a specific variable, select it on the Variables pane of the Debug tool window, then choose Run | Evaluate Expression or press `Alt+F8`.
2. In the Evaluate Expression dialog box, specify the expression you want to evaluate. Do one of the following:
   - In the Expression field, type the expression in question or choose one of the previously evaluated expressions from the drop-down list.
     If you have selected a specific variable on the Variables pane, this variable will be displayed in the Expression text box.
   - To evaluate a code fragment, click the Code Fragment Mode button and fill in the Code Fragment text box.
     To return to the original mode, click the Expression mode button.
3. Click the Evaluate button. The Result read-only field shows the evaluation output. If the specified expression cannot be evaluated, the Result field explains the reason.

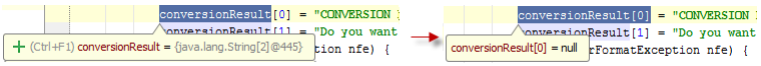
Evaluating expressions in the editor

During a debugger session, the value of any expression is shown in the tooltip every time you hover your mouse pointer over it. If an expression contains children, clicking + expands the node and displays all children.

You can also use the Quick Evaluate expression functionality that lets you view the value of an expression using the keyboard only.

There are two ways to evaluate an expression quickly:

1. By using the Show value tooltip on code selection functionality:
   a. In the Debugger | Data Views settings page, enable the Show value tooltip on code selection option.
   b. Select a code fragment with the mouse, or by clicking `Ctrl+W`. A tooltip with the expression value automatically appears under the selection and changes each time you change the selection:



2. By manually invoking the tooltip with the expression value:
   a. Place the caret at the desired location, or select an expression to be evaluated.
   b. Choose Run | Quick Evaluate Expression on the main menu, or press `Ctrl+Alt+F8`. The tooltip with the expression value appears under the selected expression.

Evaluating variables in the console pane

In the PHP and JavaScript context, you can view the values of variables right in the Debug tool window during a debugging session. This can be done in several ways:

- Select the variable in question in the Variables pane and choose Evaluate in Console on the context menu of the selection. When you switch to the Console pane, the variable name is shown in green at `>` and its value is displayed below in blue.
- Alternatively you can type the name of any variable at `>` in the Console manually and press `Enter` to have its value displayed. Code completion is at your disposal: as you type the name of a variable, PhpStorm displays a suggestion list. Note that this functionality is available only when the Use Console Input toggle button is pressed.
- To evaluate a previously evaluated variable without searching for it in the Variables pane, find the variable in question using the Up and Down arrows on you keyboard and press `Enter`.

Executing code in the console pane during a debugging session

During a PHP debugging session, you can not only evaluate variables, but also change their values, call PHP functions, and define additional ones right in the Console pane. Note that this functionality is not available for PHPUnit or Behat run/debug configurations.

1. Start a debugging session (see PHP Debugging Session) and switch to the Console pane of the Debugger tab.
2. Press the Use Console Input toggle button on the toolbar.
3. Type a statement or expression at `>` and press `Enter`. PhpStorm evaluates your code fragment and shows the output below the input code. You can type most PHP constructs including class declarations, function declarations, variables, expressions, etc. Code completion is at your disposal: as you type, PhpStorm displays a suggestion list.

4. When typing a multi-line code fragment, press `Shift+Enter` to start a new line and `Ctrl+Enter` to split a line.
5. Output for arrays and objects is by default wrapped in a `var_export()` function and displayed in the Console. To hide the output displayed, clear the Show array and object children in Debug Console check box on the Debug page.

To navigate to the source code, do one of the following:

- Select the desired item in the Variables tab and press `F4`.
- Right-click an item in the Variables tab, and select Jump to Source from the context menu.

Stepping Through the Program

In this section:

- Stepping Through the Program
  - Introduction
  - Stepping through the program
  - Suppressing stepping into
  - Tips and tricks
- Choosing a Method to Step Into

Introduction

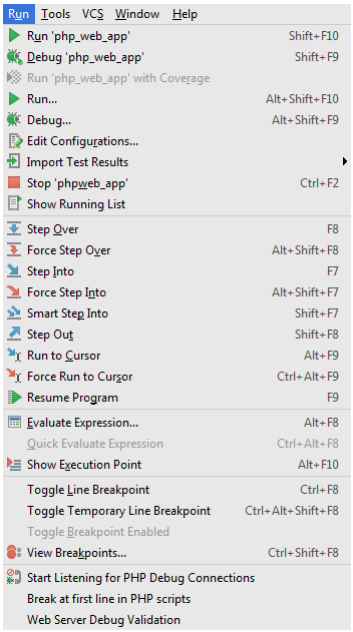
When a breakpoint is reached the Debug tool window becomes active and enables you to get control over the program's execution. For this purpose, you can use the Run menu commands, or the icons on the stepping toolbar of in the Debug tool window.

Each stepping action advances the execution point to the next execution location, depending on the action you choose.

Stepping through the program

Do one of the following:

- On the main menu, choose Run | <stepping command>



- Use the keyboard shortcuts .
- Use the buttons in the stepping toolbar of the Debug tool window.

Suppressing stepping into

You can create a list of scripts, methods, and functions that you want the debugger skip while stepping through the program.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Buld, Execution, Deployment node, and then click page_name under Debugger.
2. On the Stepping page that opens:
   - To have the debugger skip all library scripts, select the Do now step into library scripts check box.
   - To suppress stepping into some specific scripts, select the Do not step into scripts check box and create a list of such scripts using the toolbar buttons.
3. To have some PHP methods skipped, specify them on the Step Filters page:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. click Step Filters under Debug.
   2. On the Step Filters page that opens, select the check boxes next to the relevant method types: Magic methods and Constructors.
   3. If necessary, add specific methods to the Skipped Methods list using the + toolbar button.

Tips and tricks

- The Force Step Into command enables you to step into a method of a class not to be stepped into .

  The classes, stepping into which is suppressed, are specified on the Stepping page of the Settings/Preferences dialog box.
- The Force Step Over command enables you to jump over the method call ignoring the breakpoints on the way.
- The Force Run to Cursor command enables you to jump to the cursor position ignoring existing breakpoints on the way.

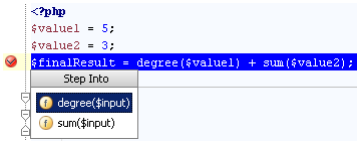
Choosing a Method to Step Into

When you reach a line with calls of several methods, you can choose the method you want to step into.

To choose a method to step into

1. On the main menu, choose Run | Smart Step Into or press `Shift+F7`.

2. In the pop-up window, choose the desired method from the list.



Finding the Current Execution Point

When a program is suspended, the source file, associated with the current execution point, is opened in the editor. The current execution point (the next line to be executed) is marked with a blue line.

You can visit the other files, and then return to the current execution point using the actions described in this section.

To find the current execution point, do one of the following

- On the main menu, choose Run | Show Execution Point.
- Press `Alt+F10`.
- Click on the stepping toolbar of the Debug tool window.

Monitoring the Debug Information

Information of a debugging session is displayed in the dedicated tabs of the Debug tool window, named after the selected run/debug configuration.

For each session, use the Console tab to view the debugger messages and application output, and the Debug tab to monitor threads and frames.

When displaying and modifying local variables or watches values, PhpStorm uses the Default Encoding setting for the current project or the IDE encoding if no encoding is specified at the project level. The same setting is used when showing the PHP console script output.

Live Editing of HTML, CSS, and JavaScript

On this page:

- Introduction
- Before you start
- Configuring the Live Edit update policy in the debugging mode
- Live Editing during a debugging session

Introduction

During a JavaScript debugging session, you can monitor how the changes you make in the HTML file are rendered in the actual browser window without refreshing the page manually and even without leaving PhpStorm. All the changes you make are immediately reflected in the browser and the affected area is highlighted. Besides pure HTML files, this also works for other file types that contain or generate HTML, CSS, or JavaScript. You just need to specify the URL of the page in the `JavaScript Debug` run configuration. The live contents of the page being edited are shown in the `Elements` tab of the Debug tool window .

Currently the Live Edit functionality is supported only for Google Chrome and only during a debugging session.

Before you start

1. Install and enable the LiveEdit plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Make sure the JetBrains Chrome Extension is installed in your Chrome browser.
   The extension is installed when you first initiate a debugging session and follow the dedicated link provided by PhpStorm. For details, see Using JetBrains Chrome Extension .

Configuring the Live Edit update policy in the debugging mode

In the debugging mode, PhpStorm provides the Live Edit functionality which supports both automatic and manual upload of updated files on the server side. For the client side, Live Edit does not provide any way to apply the changes.

In Meteor projects, the client-side code can be updated through the native Meteor hot code pushes functionality . See Using Meteor for details.

To configure the Live Edit update policy in the debugging mode

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Debugger node under Build, Execution, Deployment, and then click Live Edit. The Live Edit page opens.

2. In the Update area, configure the way changes made to the code during a debugging session are applied. Note that an update will now be performed only if none of the modified files have any syntax errors.
   - Auto in (ms): Choose this option to have the changes applied automatically at certain time interval and specify the delay in ms in the text box. This policy is not available for the client-side code of Meteor applications.
   - Manual: Choose this option to apply the changes manually by clicking the Update <run configuration name> JavaScript button or the Update <run configuration name> button on the toolbar of the Debug tool window.
     It is recommended that you choose this option for debugging Meteor applications because applying changes to the client-side code is not supported.
   - Restart if hotswap fails:
     With changes in HTML, CSS, and JavaScript on the client side, the contents of a Web page in the browser are updated without reloading. For NodeJS or Meteor applications, PhpStorm first tries to update the application incorporating the changes without restarting the NodeJS server.
     - Select this check box to have PhpStorm try restarting the server if the changes cannot be applied automatically. After the restart, PhpStorm brings you to the execution point where the problem took place.
       If even with this option chosen automatic upload still fails, tou will have to restart the server manually by clicking the Rerun <run configuration name> button .
     - When the check box is cleared, PhpStorm just displays a pop-up window informing you about the failure and suggesting you to restart the server manually.

Live Editing during a debugging session

Live Editing is enabled automatically, no steps are required from your side to activate it.

To perform Live Editing during a debugging session

1. When editing the code and monitoring how the changes are rendered in the browser, it is helpful to have the areas affected by the changes highlighted. To enable highlighting affected areas, open the PhpStorm settings dialog box, click Live Edit, and then select the Highlight current element in browser check box on the Live Edit page that opens.

2. To initiate a debugging session, create a run configuration of the type JavaScript Debug and click the Debug button on the toolbar. PhpStorm establishes connection with the JetBrains Chrome extension and starts a debugging session.

3. Edit the HTML file that implements the page opened in the browser and view the changes immediately rendered.

PhpStorm supports an operation that is somehow opposite to live editing. During a debugging session, the Elements tab of the Debug tool window shows the HTML source code that implements the actual browser page and its HTML DOM structure . Moreover, any changes made to the page through the browser are immediately reflected in the Elements tab. For more details, see Viewing Actual HTML DOM .

Viewing Actual HTML DOM

On this page:

- Introduction
- Before you start
- Viewing the HTML source and DOM structure

Introduction

Besides Live Editing, that is, monitoring how the changes you make in an HTML file are rendered in the actual browser window, during a debugging session you can perform a reverse operation using the Elements tab of the Debug tool window. The tab shows the HTML source code that implements the actual browser page and its HTML DOM structure . Moreover, any changes made to the page through the browser are immediately reflected in the Elements tab.

Before you start

1. Make sure you are using the Chrome browser for debugging.
2. Make sure the JetBrains Chrome Extension is installed in your Chrome browser.
   The extension is installed when you first initiate a debugging session and follow the dedicated link provided by PhpStorm. For details, see Using JetBrains Chrome Extension .
3. Make sure the LiveEdit repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Viewing the HTML source and DOM structure

To view the HTML source and DOM structure of the actual page

1. To initiate a debugging session, create a run configuration of the type JavaScript Debug and click the Debug button on the toolbar. PhpStorm establishes connection with the JetBrains Chrome extension and starts a debugging session.

2. Switch to the Debug tool window and open the Elements tab. The tab consists of three panes, all the panes are read-only. The Text pane shows the HTML source code of the page that is currently opened in the browser. As soon as any change is made to the page in the browser (e.g. clicking an icon), the code in the pane is updated accordingly.
   The Structure pane shows the DOM structure of the HTML code in the Text pane.

   The Scripts pane shows a tree of executed scripts.

   The Structure and Text panes are mutually synchronized. When you click a node in the DOM structure, PhpStorm scrolls through the contents of the Text pane.

   The panes are also synchronized with the browser. PhpStorm highlights the element in the browser as soon as you click the corresponding node in the DOM structure or in the Text pane.

Using JetBrains Chrome Extension

On this page:

- Introduction
- Installing the JetBrains Chrome extension
- Working with JetBrains Chrome extension
- Overriding the default CORS settings
- Changing port

Introduction

The JetBrains Chrome extension is mainly responsible for debugging JavaScript in Chrome and Dartium. During a debugging session, the extension also supports Live Editing of HTML, CSS, and other code that contains or produces JavaScript, and provides actual HTML DOM and dynamically updated source code of the actual page.

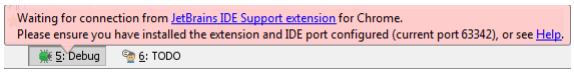
With the JetBrains Chrome extension, you can run several debugging sessions controlled by several IDEs and switch between them by changing the port which the extension listens to. And finally, the extension provides the possibility to override default CORS (Access-Control_Allow_Origin) settings and access local scripts on your machine that are by default protected.

Installing the JetBrains Chrome extension

You can install the extension during the first debugging session with Chrome or Dartium or at any time by visiting the Chrome Web Store.

To install the JetBrains Chrome extension

1. Do one of the following:
   - At any time open the Chrome Web Store at https://chrome.google.com/webstore/detail/jetbrains-ide-support/hmhgeddbohgjknpmjagkdomcpobmllji .
   - Start a JavaScript debugging session: create a run configuration with Chrome as the debugging browser, and click the Debug button on the toolbar. For details, see Debugging JavaScript . PhpStorm informs you that it is waiting for connection with the JetBrains IDE Support extension and shows the following message in the Debug tool window:

     Click the JetBrains IDE Support extension link which brings you to the Chrome Web Store.

   In either case the JetBrains IDE Support page opens where you can learn more about the extension.
2. Click the Add to Chrome button

.
3. In the Confirm New Extension dialog box that opens, click Add. The Add to Chrome button changes to Added to Chrome

.

When the extension is installed, the icon is displayed next to the Chrome address bar.

Working with JetBrains Chrome extension

Control over the JetBrains Chrome extension is provided through the chrome://extensions/ page.

To enable, disable, or remove the JetBrains Chrome extension

- To open the chrome://extensions/ page, click the Customize and control Chromium button next to the address bar, then choose Tools | Extensions on the context menu.
- To deactivate the extension, clear the Enabled check box. The check box name changes to Enable.
- To activate the extension, select the Enable check box.
- To uninstall the extension, click the Remove from Chromium button.

Overriding the default CORS settings

Suppose the page you are debugging requests a resource which is protected against access for security reasons through CORS settings. See more about CORS at https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS .

You can enable access to the resources protected by default by changing the Chrome extension options.

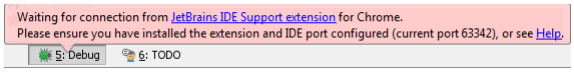
To override the default CORS settings

1. Right-click and choose Options on the context menu. A web page with Chrome extension options opens showing the parameters to connect to PhpStorm.
2. In the Force CORS text box, type the pattern that defines the URL addresses you want to make accessible, for example: `http://youtrack.jetbrains.com/rest/*` .

Changing port

During a debugging session, the Chrome extension listens to the port of the software product from which the extension was invoked. Each software product including PhpStorm has its own default port on which it starts by default.

If for some reason the default PhpStorm port is already busy, PhpStorm finds the closest available port and starts on it. This results in a conflict: PhpStorm is running on a "new" port while the Chrome extension still listens to the port of a previously started product and fails to establish connection with PhpStorm. The conflict reveals when you initiate a debugging session from PhpStorm: the extension fails to connect through the default port, PhpStorm waits for connection from the extension and displays the following message with the port number where it is actually running:



To fix the problem, specify the actual PhpStorm port in the Chrome extension options.

To change the port for connecting to PhpStorm

1. Right-click and choose Options on the context menu. A web page with Chrome extension options opens showing the parameters to connect to PhpStorm.

2. In the IDE Connection area, specify the actual PhpStorm port in the Port spin box.

Inline Debugging

On this page:

- Basics
- Enabling inline debugging
- Viewing inline debugging results

Basics

The inline debugging functionality facilitates the debugging procedure, as it lets you view the value of variables used in your source code right next to their usage, without having to switch to the Variables pane of the Debug tool window .

Enabling inline debugging

To enable the inline debugging functionality, do one of the following:

- In the Debug tool window toolbar, click the Settings icon and select the Show Values Inline option from the popup menu.
- Open the page of Setting/Preferences dialog, and select the check box Show values inline.

Viewing inline debugging results

If this option is enabled, when you launch a debug session and step through the program, the values of variables are displayed at the end of the lines where these variables are used.

```
17    while ( $i < 10 ) {
18        echo "The result is " . $i . "<br> ";
19        $i = $i + 1;   $i: 3
```

Testing

This section covers the issues that are common for all the supported testing frameworks:

- Creating Run/Debug Configuration for Tests
- Performing Tests
- Monitoring and Managing Tests
- Rerunning Tests
- Terminating Tests
- Viewing and Exploring Test Results

For the framework-specific testing procedures, refer to the sections:

- Testing PHP Applications
- Unit Testing JavaScript

To create and run unit tests on PHP applications, perform the following general steps:

- Enable PHPUnit support .
- Write unit tests and mark the folder where they are stored as test folder .
- Create a run configuration :
  - To run unit tests locally, create a PHPUnit configuration.
  - To run unit tests on a remote server, create a PHPUnit by HTTP configuration.
- Launch unit tests and monitor test results in the Run tool window.

In this part:

- Creating Run/Debug Configuration for Tests
- Performing Tests
- Monitoring and Managing Tests
- Rerunning Tests
- Terminating Tests
- Viewing and Exploring Test Results

Creating Run/Debug Configuration for Tests

On this page:

- Introduction
- Creating a test configuration
- Creating a test configuration for a specific target

Introduction

You can run your tests (test cases, test suites, etc.) using run/debug configurations, in the way similar to running ordinary applications. PhpStorm provides a framework for creating special run/debug configurations for testing purposes, where a test can be specified as a target.

In addition to the regular test configuration creation , PhpStorm supports creating test configurations for specific targets , for example for all tests in a file, class, or directory.

With PhpStorm you can also create configurations for running PHP unit tests locally or on a remote server.

Creating a test configuration

To create a test configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Press `Shift+Alt+F10` and choose Edit Configuration on the context menu.

2. Click + on the toolbar and select the desired configuration type:
   - To have unit tests executed locally, choose PHPUnit.
   - For remote test execution, choose PHPUnit by HTTP.
   - JSTestDriver for testing JavaScript using JSTestDriver. See Running Unit Test on JSTestDriver and Run/Debug Configuration: JSTestDriver .
   - Karma for testing JavaScript using JSTestDriver. See Running Unit Tests on Karma and Run/Debug Configuration: Karma .
   - NodeUnit for testing NodeJS. See Running Nodeunit Tests and Run/Debug Configuration: NodeUnit .
   - DartUnit for testing Dart, see Run/Debug Configuration: Dart Test .

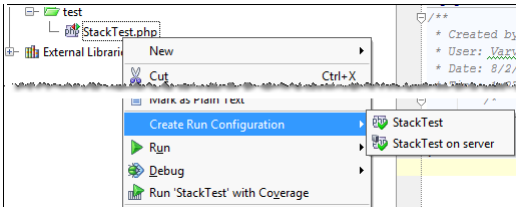
3. In the dialog box that opens, specify the test scope, configuration parameters, and activities to perform before test execution. Apply the changes and close the dialog box.

Creating a test configuration for a specific target

To create a test configuration for a specific target

1. In the Project tool window, right-click the desired test directory, individual file, or class.

2. On the context menu of the selection, choose Create Run configuration. PhpStorm displays a list of suggested configurations.



   Each configuration has the name of the selected target and is supplied with an icon indicating its type.
   - - the icon marks a PHPUnit configuration for running tests locally.
   - - the icon marks a PHPUnit by HTTP configuration for running tests remotely.

   Choose the configuration of the desired type.

3. In the Run/Debug Configuration dialog box that opens, specify the configuration parameters and activities to perform before test execution. Apply the changes and close the dialog box.

Performing Tests

On this page:

– Introduction

– Running or debugging a test

Introduction

Generally, PhpStorm runs and debugs tests in the same way as other applications, by running the run/debug configurations you have created . When doing so, it passes the specified test classes or methods to the test runner.

In many cases, you can initiate a testing session from a context menu. For this purpose, the Run and Debug commands are provided in certain context menus. For example, these commands are available for a test class, directory, or a package in the Project tool window . They are also available for a test class or method you are currently working on in the editor.

If you run a test for which there is no permanent run/debug configuration a temporary configuration is created. You can then save such a configuration using the Run/debug configuration dialog, if you want to reuse it later.

The tests run in the background, so you can execute several tests at the same time.

Each running configuration gets its own tab in the Run tool window (the Test Results tab). One tab can aggregate several tests.

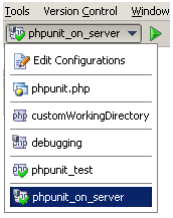
Running or debugging a test

To run or debug a test, follow these general steps:

To start running or debugging a test, you can use the main toolbar, or a context menu in the Project tool window or in the editor:

– Using the main toolbar:

1. Select the necessary run/debug configuration from the list on the main toolbar.



2. Click Run or Debug to the right of the list. (Alternatively, choose Run | Run (`Shift+F10`) or Run | Debug (`Shift+F9`) from the main menu.)

> **Tip** To see the list of available run or debug configurations and to quickly select the one you want, you can use the following keyboard shortcuts: `Shift+Alt+F10` for the run configurations, or `Shift+Alt+F9` for the debug configurations.

– Using a context menu:

Right-click a test file or test class in the Project tool window, or open it in the editor, and right-click the background. On the context menu, choose Run < class name> / Run < file name> or Debug....

> **Tip** For a test method, open the class in the editor and right click anywhere in the method. The context menu suggests the command Run / Debug <method name>.

Monitoring and Managing Tests

Test progress and results display in the dedicated test runner tabs of the Run tool window.

You can rerun , terminate, and suspend execution of tests same way as you do it for running applications. In addition to the common running actions, in the test runner you can:

- Navigate through the list of test cases using arrow keys.
- Navigate between failed tests using the and buttons or `Ctrl+Alt+Up` or `Ctrl+Alt+Down` keyboard shortcuts.
- View the total number of tests being run in the current session. The summary information is displayed in a message line on the top of the tool window. After completion of the tests, the message informs you about the number of failed tests and elapsed time.
- View testing progress in the progress bar.
- Show or hide information about the passed tests, using the button.
- Show the ignored tests in the tree view of all tests within the current run/debug configuration or test class using the button.
- Navigate from the stack trace to the problem location in the source code by clicking the hyperlink in the Output pane.
- Enable and disable the following functionality by clicking the cog button and selecting the relevant items from the context menu:
  - Monitoring execution of the current test.
  - Have the first failed test selected automatically upon completing the suit.
  - Navigating to the stack trace.
  - Automatic scrolling to the source code.
  - Have the corresponding source code opened at exceptions.
  - Have statistic shown in the Statistics pane.

Rerunning Tests

On this page:

- Introduction
- Rerunning a testing session
- Rerunning a failed test

Introduction

You can repeat your test session without leaving your test runner tab of the Run tool window. The tests are performed again using the same run configuration as in the initial run.

Rerunning a testing session

Do one of the following:

- Click the rerun button on the toolbar of the Run tool window.
- Use `Ctrl+F5` keyboard shortcut.

Rerunning a failed test

1. In the testing tab of the test runner, select a failed test.
2. In the Run toolbar , click Rerun Failed Tests .

Terminating Tests

You can abort a running test session at any time. So doing, all tests that are current running will stop immediately. The icons of the tests in the test runner reflect the statuses of the tests (passed, failed, aborted, or never run).

To stop a testing session, do one of the following

- On the toolbar of the test runner, click the stop button .
- Use `Ctrl+F2` keyboard shortcut.

Viewing and Exploring Test Results

On this page:

- Overview
- Viewing statistics

Overview

Depending on the selected node in the tree view of tests, the Test Runner displays information on the various levels. In the Test Runner, you can view statistics of the tests, navigate to stacktrace, show or hide successful tests, and more.

Use the Testing toolbar to control visual representation of the test results.

Viewing statistics

The Statistics tab shows information about the elapsed time and memory usage of each test.

To view statistics:

- Click on the Testing toolbar to reveal the drop-down menu, and then click the Show Statistics check command.

The values displayed in the Statistics tab are not accurate and only give an approximate estimate of the test performance. For example, if a garbage collector works during the test run, the memory usage shown in the Statistics tab is wrong.

Code Coverage

Basics

Code coverage in PhpStorm allows you to perform on-the-fly line coverage measuring for your code with low runtime overhead. In general, line coverage answers the question, "Was this line of code executed during unit testing simulation?"

Measuring code coverage is available for PHPUnit tests or JSTestDriver unit tests .
Code coverage results are reflected in the dedicated Coverage tool window , in the Project view of the Project tool window , and in the editor. The tool windows show the following information:

- For a directory: the percentage of the covered classes and lines.
- For a file: the percentage of the covered lines.

When a file is opened in the editor, each line is highlighted with regard to its code coverage status:

- Lines executed during simulation are marked green.
- Lines not executed during simulation are marked red.

The coverage measurement results comprise a coverage suite. You can have the results of a new simulation merged with any existing suite. In this case, a line will be considered covered if it is covered by at least one of the simulations.

A coverage suite is generated every time a test or application with code coverage measurement is executed. It is possible to have an unlimited amount of coverage suites.

To configure code coverage behavior

1. Open the Settings dialog box , and then click Coverage under Build, Execution, Deployment. The Coverage page opens.

2. Define how the collected coverage data will be processed:
   - To have the Code Coverage dialog box shown every time you launch a new run configuration with code coverage, choose Show options before applying coverage to the editor.
   - To discard the new code coverage results, choose Do not apply collected coverage.
   - To discard the active suites and use the new one every time you launch a new run configuration with code coverage, choose Replace active suites with the new one.
   - To have the new code coverage suite appended to the active suites every time you launch a new run configuration with code coverage choose Add to active suites.

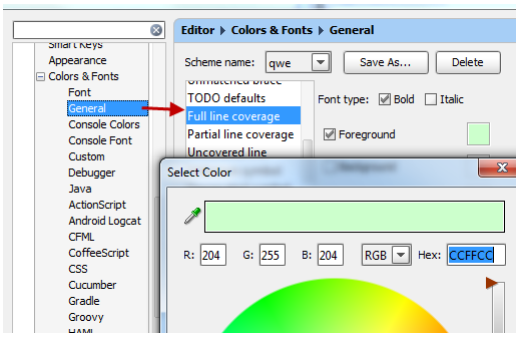
3. Define the behaviour of the Coverage tool window when an application or test is run with coverage:
   - To have the Coverage tool window opened automatically, select the Activate Coverage View check box.
   - To open the Coverage tool window manually, clear the Activate Coverage View check box.

To configure code coverage colors

1. Open the Editor - Colors and Fonts page of the Settings dialog.
   Alternatively, just click in the statistics pop-up.

2. Expand Colors and Fonts node, and select General.

3. In the list of textual components, select the required type of coverage, for example, Full line coverage or Uncovered line, and then choose the desired colors:



Running with Coverage

PhpStorm provides a dedicated action that allows you to run tests with code coverage measurement. The code coverage data are processed according to the option selected in the Coverage page of the Settings dialog box.

To run a test with code coverage measurement

1. Do one of the following:
   - Open the desired file in the editor, and choose Run <name> with coverage on the context menu. When running tests with coverage, note that you can run the entire test class, or each individual test method, depending on the caret location.
   - Select the desired run/debug configuration, and then on the main menu choose Run | Run <run/debug configuration name> with coverage.
   - On the main toolbar, click . This will launch the selected run/debug configuration.

2. If the Show options before applying coverage to the editor check box has been selected in the Coverage page of the Settings dialog box, choose whether you want to replace the active coverage suites, or add the collected data to the active suites, or do not want not apply coverage data. You can also opt to skip this dialog in the future.
   In case any other option has been selected, the respective action will be performed silently.

3. Explore the collected coverage data in the Coverage tool window .

Viewing Code Coverage Results

Viewing code coverage helps you detect pieces of your source code that are not affected by simulation.

To view code coverage results

1. Create a testing run/debug configuration , select it in the Run/Debug Configurations drop-down list, and click the Run with Coverage button .
2. Do one of the following:
   - On the main menu, choose Tools | Show Code Coverage Data.
   - Press `Ctrl+Alt+F6`.
3. In the Choose Coverage Suite to Display dialog box, select check boxes next to the suites to view and click the Show selected button.
4. View coverage results in the dedicated Coverage tool window.
5. Create missing tests. For JavaScript, tests can be generated as described in section Creating JavaScript Unit Tests .

Working with Web Servers: Copying Files

In this section:

- Working with Web Servers: Copying Files
  - Basics
  - Interaction between PhpStorm and servers
- Configuring Synchronization with a Web Server
- Uploading and Downloading Files
- Accessing Files on Web Servers
- Comparing Deployed Files and Folders with Their Local Versions
- Editing Individual Files on Remote Hosts

Basics

Among numerous ways to configure your development and production environments the most frequent ones are as follows:

- The Web server is installed on your computer. The sources are under the server document root (for example, `/htdocs`), and you do your development right on the server.
- The Web server is installed on your computer but the sources are stored in another folder. You do your development, then copy the sources to the server.
- The Web server is on another computer (remote host). Files on the server are available through the FTP/SFTP/FTPS protocol, through a network share, or a mounted drive.

Now let's see how to use PhpStorm in the above environment configurations. PhpStorm assumes that all development, debugging, and testing is done on your computer and then the code is deployed to a production environment.

Please note the following:

- The reason to stick to this "local development - deployment" model lies in the way PhpStorm provides its coding assistance which includes code completion, code inspections & validations, code navigation, etc. All this functionality is based on the index of the project files which PhpStorm builds when the project is loaded and updates on the fly as you edit your code.
- To provide efficient coding assistance, PhpStorm needs to re-index code fast, which requires fast access to project files. The latter can be ensured only for local files, that is, files that are stored on you hard disk and are accessible through the file system. Therefore PhpStorm does not support the mode when you access your files over a network folder (very often it becomes slow and unresponsive, performs random look-ups for no obvious reason, etc).

Interaction between PhpStorm and servers

Interaction between PhpStorm and servers is controlled through server access configurations. Anytime you are going to use a server, you need to define a server access configurations, no matter whether your server is on a remote host or on your computer.

Taking into account all the above, let's define the following basic concepts related to synchronization between PhpStorm and servers.

- An in-place server is a server whose document root is the parent of the project root, either immediate or not. In other words, the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.
- A local server is a server that is running in a local or a mounted folder and whose document root is NOT the parent of the project root.
- A remote server is a server on another computer (remote host).
- The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.
- A local file/folder is any file or folder under the project root.
- A remote file/folder is any file or folder on the server, either local or remote.
  Suppose you have a project `C:/Projects/My_Project/` with a folder `C:/Projects/My_Project/My_Folder` and a local server with the document root in `C:/xampp/htdocs`. You upload the entire project tree to `C:/xampp/htdocs/My_Project`. In the terms of PhpStorm, the folder `C:/Projects/My_Project/My_Folder` is referred to as local and the folder `C:/xampp/htdocs/My_Project/My_Folder` is referred to as remote.
- Upload is copying data from the project TO the server, either local or remote.
- Download is copying data FROM the server to the project.

After you have configured synchronization with a server, you can upload, download, and manage files on it directly from PhpStorm. Moreover, you can suppress uploading or downloading specific files or entire folders. Finally, you can optimize you workflow by configuring content roots so specific folders are not involved in indexing, which significantly saves project indexing time.

Synchronization with servers, uploading, downloading, and managing files on them are provided via the Remote Hosts Access bundled plugin, which is by default enabled. If the plugin is disabled, activate it in the Plugins page of the Settings dialog box. For details, see Enabling and Disabling Plugins .

Configuring Synchronization with a Web Server

In this section:

- Configuring Synchronization with a Web Server
  - Before you start
  - Basics
    - Server access configuration
- Creating an in-Place Server Configuration
- Creating a Local Server Configuration
- Creating a Remote Server Configuration
- Customizing Upload/Download
- Excluding Files and Folders from Upload/Download

Before you start

Synchronization with servers, uploading, downloading, and managing files on them are provided via the Remote Hosts Access bundled plugin, which is by default enabled. If the plugin is disabled, activate it in the Plugins page of the Settings dialog box. For details, see Enabling and Disabling Plugins .

Basics

PhpStorm distinguishes among in-place, local, and remote servers, however the meaning of these terms in the context of PhpStorm slightly differs from their common meaning:

- An in-place server is a server whose document root is the parent of the project root, either immediate or not. In other words, the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.
- A local server is a server that is running in a local or a mounted folder and whose document root is NOT the parent of the project root.
- A remote server is a server on another computer (remote host).

For more information about possible configuration of the production and development environment and working with servers from PhpStorm, see Working with Web Servers: Copying Files .

Server access configuration

PhpStorm controls interaction with Web servers through server access configurations. Anytime you are going to use a server, you need to define a server access configurations, no matter whether your server is on a remote host or on your machine.

A server access configuration defines:

- The server type (in-place, local, or remote).
- The computer (host) where the server is running. For in-place and local servers, PhpStorm presupposes that it is the current computer where your project is.
- The server access configuration root: the highest folder in the server hierarchy that can be accessed through the server configuration.
- The URL address to access the server configuration root. Both the HTTP and the HTTPS protocols are supported.
  To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts` . Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts` .

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit` .
3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

   Learn more at Java6 and Java7 .

- The protocol to transfer the data through.
- Correspondence between local (project) folders, destination folders on the server, and URL addresses to access the data on the server. This correspondence is called mapping.

You can define as many configurations as necessary, thus enabling flexible switching between upload/download setups.

Creating an In-Place Server Configuration

On this page:

- Basics
- Creating a server configuration: specifying its name, type, and visibility
- Configuring access to an in-place server: specifying the URL address of the server document root
- Specifying the project root folder and the URL address to access it

Basics

An in-place server is a server whose document root is the parent of the project root, either immediate or not. In other words, the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.

To configure access to the server in this set-up, you only need to specify the URL address of the sever document root, appoint the project root folder, and specify the URL address to access it.

Creating a server configuration: specifying its name, type, and visibility

1. Open the Deployment page by doing one of the following:
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
   - Choose Tools | Deployment | Configuration on the main menu.
2. In the left-hand pane, that shows a list of all the existing server configurations, click the Add toolbar button +. The Add Server dialog box opens.
3. Specify the server configuration name in the Name text box. From the Type drop-down list, choose the server configuration type In-place. Use the Up and Down keyboard keys to scroll through the list of server configuration types.
4. Use the Visible only for this project check box to configure the visibility of the server access configuration (deployment configuration).
   - Select the check box to restrict the use of the configuration to the current project. Such configurations cannot be reused outside the current project, they do not appear in the list of available configurations in other projects.
   - When the check box is cleared, the configuration is visible in all PhpStorm projects and the settings from, including SSH credentials, can be reused.
5. Click OK. The Add Server dialog box closes and you return to the Connection tab of the Deployment node.

Configuring access to an in-place server: specifying the URL address of the server document root

In the Web server root URL text box, type the URL address associated with the document root of your Web server as defined in the Web server configuration file. This URL address will be the starting point for building the URL address of your application. Both the HTTP and the HTTPS protocols are supported.

To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_na
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`. Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.
3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7.

For example, the Apache httpd server configuration file is `httpd.conf`. The default document root is the `htdocs` folder and the default URL address to access the data in it is `http://localhost`. If you have changed the default port `80`, you have to specify the port explicitly: `http://localhost:<port>`.

Specifying the project root folder and the URL address to access it

1. Switch to the Mappings tab.
2. In the Local path text box, specify the full path to your project root folder. Type the path manually, or click the Browse button and choose the folder in the dialog box, that opens.
3. In the Web path on server text box, type the path to the project root folder relative to the server document root specified in the server configuration file. As you type, PhpStorm dynamically builds the URL address through which your project root folder will be accessible and shows it as a link in the Project URL read-only field. To check that the URL address is constructed correctly and ensures access to the project root, click the link.

Creating a Local Server Configuration

On this page:

- Basics
- Creating a server configuration: specifying its name, type, and visibility
- Specifying the server configuration root and the URL address to access it
- Example of specifying a server configuration root
- Mapping project folders with folders on the server and the URL addresses to access them
- Example of mapping project folders with folders on the server

Basics

A local server is a server that is running in a local or a mounted folder and whose document root is NOT the parent of the project root.

To configure access to the server in this set-up, you need to specify the following:

1. The server configuration root folder and the URL address to access it.
2. Correspondence between the project root folder, the folder on the server to copy the data from the project root folder to, and the URL address to access the copied data on the server. This correspondence is called mapping.

Creating a server configuration: specifying its name, type, and visibility

1. Open the Deployment page by doing one of the following:
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
   - Choose Tools | Deployment | Configuration on the main menu.
2. In the left-hand pane, that shows a list of all the existing server configurations, click the Add toolbar button +. The Add Server dialog box opens.
3. Specify the server configuration name in the Name text box. From the Type drop-down list, choose the server configuration type Local or mounted folder.
   When editing the server configuration name in the Name text box, use the Up and Down keys on your keyboard to change the preselected server access to type in the Type drop-down list.
4. Use the Visible only for this project check box to configure the visibility of the server access configuration (deployment configuration).
   - Select the check box to restrict the use of the configuration to the current project. Such configurations cannot be reused outside the current project, they do not appear in the list of available configurations in other projects.
   - When the check box is cleared, the configuration is visible in all PhpStorm projects and the settings from, including SSH credentials, can be reused.
5. Click OK. The Add Server dialog box closes and you return to the Connection tab of the Deployment node.
   Click the Use as Default toolbar button to have PhpStorm silently apply the current configuration in the following cases:
   - Automatic upload of changed files .
   - Manual upload of files without choosing the target host.
   - Comparing local files and folders with their remote versions.

Specifying the server configuration root and the URL address to access it

1. In the Folder text box of the Upload/download project files area, specify the server configuration root.
   The server configuration root is the highest folder in the file tree on the server that can be accessed through the server configuration. The easiest way is to use the document root of your Web server as defined in the Web server configuration file. However you can appoint any other existing folder under the document root.
2. In the Web server root URL text box of the Browse files on server area, specify the URL address of the server configuration root. This URL address will be the starting point for building the URL address of your application. Depending on your choice of the server configuration root, do one of the following:
   - Type the URL address associated with the document root of your Web server as defined in the Web server configuration file.
   - Type the URL address in the format `<URL of the server document root>/<path to the relevant folder relative to the server document root>`.

   Both the HTTP and the HTTPS protocols are supported.
   To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`.
   Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.
3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

Example of specifying a server configuration root

For example, the Apache httpd server configuration file is `httpd.conf`, according to it, the default document root is the `htdocs` folder, and the default URL address to access the data in it is `http://localhost`. For the sake of simplicity, let's suppose that you are using the XAMPP package and it is installed in the root of the `C:/` drive.

So if you decide to copy your project files directly under the server document root, your server configuration root will be `C:\xampp\htdocs` and its URL will be `http://localhost:<port>`.

You can establish a more complicated folder structure on the server, for example, to have `MySite1` and `MySite2` folders under the server document root. In this case the you will have to decide which of these folders you will use in the current configuration, let it be `MySite2`. Accordingly, the server configuration root will be `C:\xampp\htdocs\MySite2` and its URL address will be `http://localhost:<port>\MySite2`.

Mapping project folders with folders on the server and the URL addresses to access them

Configure mappings, that is, set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . The easiest way is to map the entire project root folder to a folder on the server, whereupon the project folder structure will be repeated on the server, provided that you have selected the Create Empty directories check box in the Options dialog box . "For more details, see Customizing Upload/Download .

1. Switch to the Mappings tab.
2. In the Local Path text box, specify the full path to the desired folder in the project tree. In the simplest case it is the project root.
3. In the Deployment Path text box, specify the folder on the server where PhpStorm will upload the data from the folder specified in the Local Path text box. Type the path to the folder relative to the server configuration root. If the folder with the specified name does not exist yet, PhpStorm will create it, provided that you have selected the Create Empty directories check box in the Options dialog box . For more details, see Customizing Upload/Download .
4. In the Web Path text box, type the path to the folder on the server relative to the server configuration root. Actually, type the relative path you typed in the Deployment Path text box.

Example of mapping project folders with folders on the server

For example, if your project is `C:\My_Projects\Mapping_project`, the server document root is `C:\xampp\htdocs`, the server configuration root is `C:\xampp\htdocs\MySite2`, and its URL address is `http://localhost:<port>\MySite2`, fill in the fields as follows:

1. In the Local Path text box, type `C:\My_Projects\Mapping_project`.
2. In the Deployment Path text box, type `Mapping_project`.
3. In the Web Path text box, type `Mapping_project`.

Creating a Remote Server Configuration

On this page:

- Basics
- Creating a server configuration: specifying its name, type, and visibility
- Specifying user credentials defined during registration on the host
- Enabling connection to the server and specifying the server configuration root
- Mapping local folders to folders on the server and the URL addresses to access them
- Overloading the deployment destination by configuring nested mappings

Basics

A remote server is a server on another computer (remote host).

To configure access to the server in this set-up, you need to specify the following:

1. Connection settings: server host, port, and user credentials.
2. The server configuration root folder and the URL address to access it.
3. Correspondence between the project root folder, the folder on the server to copy the data from the project root folder to, and the URL address to access the copied data on the server. This correspondence is called mapping.

Creating a server configuration: specifying its name, type, and visibility

1. Open the Deployment page by doing one of the following:
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
   - Choose Tools | Deployment | Configuration on the main menu.
2. In the left-hand pane, that shows a list of all the existing server configurations, click the Add toolbar button +. The Add Server dialog box opens.
3. Specify the server configuration name in the Name text box. From the Type drop-down list, choose the server configuration type depending on the protocol you are going to use to exchange the data with the server.
   - FTP: choose this option to have PhpStorm access the server via the FTP file transfer protocol .
   - SFTP: choose this option to have PhpStorm access the server via the SFTP file transfer protocol.
   - FTPS: choose this option to have PhpStorm access the server via the FTP file transfer protocol over SSL (the FTPS extension).

   When editing the server configuration name in the Name text box, use the Up and Down keys on your keyboard to change the preselected server access to type in the Type drop-down list.
4. Use the Visible only for this project check box to configure the visibility of the server access configuration (deployment configuration).
   - Select the check box to restrict the use of the configuration to the current project. Such configurations cannot be reused outside the current project, they do not appear in the list of available configurations in other projects. For example, if this check box is selected in an SFTP configuration, you cannot use your SSH credentials from it when you configure a remote interpreter.
   - When the check box is cleared, the configuration is visible in all PhpStorm projects and the settings from, including SSH credentials, can be reused.

   See Configuring Node.js Interpreters and Configuring Remote PHP Interpreters for details.
5. Click OK. The Add Server dialog box closes and you return to the Connection tab of the Deployment dialog box.
   Click the Use as Default toolbar button to have PhpStorm silently apply the current configuration in the following cases:
   - Automatic upload of changed files .
   - Manual upload of files without choosing the target host.
   - Comparing local files and folders with their remote versions.

Specifying user credentials defined during registration on the host

1. Specify the registration mode:
   - To login in a regular mode, specify the login in the User name text box.
   - To enable anonymous access to the server with your email address as password, select the Login as anonymous check box.
2. Specify the way to authenticate to the server:
   - For FTP server, type your password and select the Save password check box to have PhpStorm remember it.
   - For SFTP server, choose the way to authenticate to the server. Do one of the following:
     - To use standard authentication, choose Password, specify the password, and select the Save password check box to have PhpStorm remember it.
     - To use SSH authentication via a key pair, choose Key pair (OpenSSH).
       To apply this authentication method, you need to have your private key on the client machine and your public key on the remote server you connect to. PhpStorm supports private keys generated using the OpenSSH utility. See http://wiki.qnap.com/wiki/How_To_Set_Up_Authorized_Keys for details.

       Type the private key file and the passphrase in the corresponding text boxes.
   - For FTPS server, specify your user name and password and choose the security mechanism to apply.
     - Choose Explicit to have the explicit (active) security applied. Immediately after establishing connection, the FTP client on your machine sends a command to the server to establish secure control connection through the default FTP port.
     - Choose Implicit to have the implicit (passive) security applied. In this case, security is provided automatically upon establishing connection to the server which appoints a separate port for secure connections.

Enabling connection to the server and specifying the server configuration root

1. Specify the host name of the FTP/SFTP/FTPS server to exchange data with and the port to which this server listens. The default values are:
   - 21 for FTP and FTPS
   - 22 for SFTP
2. In the Root path text box, specify the server configuration root relative to your user home which was defined when you registered your account. This folder will be the

highest one in the folder structure accessible through the current server configuration. Do one of the following:

- Accept the default value `/`, which points at the user home folder on the server.
- Type the path manually.
- Click the Browse button ... and select the desired folder in the Choose Root Path dialog box that opens.
- Click the Autodetect button and have PhpStorm detect the user home folder settings on the FTP/SFTP server and set up the root path according to them. The button is only enabled when you have specified your user name and password.

3. In the Web server root URL text box, type the URL address to access the server configuration root (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.). Both the HTTP and the HTTPS protocols are supported. To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_
```

If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`. Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

4. Click the Open button to make sure that the specified URL address is accessible and points at the correct Web page.

Mapping local folders to folders on the server and the URL addresses to access them

Configure mappings, that is, set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . The easiest way is to map the entire project root folder to a folder on the server, whereupon the project folder structure will be repeated on the server, provided that you have selected the Create Empty directories check box in the Options dialog box . "For more details, see Customizing Upload/Download .

1. Switch to the Mappings tab.
2. In the Local Path text box, specify the full path to the desired folder in the project tree. In the simplest case it is the project root.
3. In the Deployment Path text box, specify the folder on the server where PhpStorm will upload the data from the folder specified in the Local Path text box. Type the path to the folder relative to the server configuration root. If the folder with the specified name does not exist yet, PhpStorm will create it, provided that you have selected the Create Empty directories check box in the Options dialog box . For more details, see Customizing Upload/Download .
4. In the Web Path text box, type the path to the folder on the server relative to the server configuration root. Actually, type the relative path you typed in the Deployment Path text box.

Overloading the deployment destination by configuring nested mappings

You can configure separate mappings for a specific folder under the your project root to have the contents of this folder synchronized with another location on the remote host.

Suppose you have configured the mappings as follows:

Local Path	Deployment Path
`<project_root>`	`ftp://.../htdocs/my_project`
`<project_root>/my_folder`	`ftp://.../htdocs/my_folder`

Then the files in your project will be uploaded as follows:

Local Path	Deployment Path
`<project_root>/file1.php`	`ftp://.../htdocs/my_project/file1.php`
`<project_root>/my_folder/file2.php`	`ftp://.../htdocs/my_folder/file2.php`

instead of `ftp://.../htdocs/my_project/my_folder/file2.php`

Customizing Upload/Download

On this page:

- Basics
- Setting common upload/download options
- Specifying additional protocol-specific customization options for FTP/SFTP/FTPS servers

Basics

In addition to the mandatory settings that ensure successful upload and download in various project - server set-ups, you can choose additional options to customize interaction with the server. Most of these options apply to all types of server access configuration. For FTP/FTPS/SFTP server configurations, you can specify additional protocol-specific options.

Setting common upload/download options

1. Open the Options dialog box by doing one of the following:
   - On the main menu, choose Tools | Deployment | Options.
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Build, Execution, Deployment node, and then click Options under Deployment.
2. In the Options dialog box that opens, specify additional settings:
   - To have PhpStorm skip specific files or entire folders during upload/download, in the Exclude items by name text box, specify the patterns that define the names of these files and folders. Use semicolons as delimiters. Wildcards are welcome.
     The exclusion is applied recursively. This means that if a matching folder has subfolders, the contents of these subfolders are not deployed either.

     For more details about excluding files and folders from upload/download, see Excluding Files and Folders from Upload/Download .
   - Specify the details of the upload/download procedure by selecting or clearing the corresponding check boxes.

Specifying additional protocol-specific customization options for FTP/SFTP/FTPS servers

1. Open the Deployment dialog box by doing one of the following:
   - Choose Tools | Deployment | Configuration on the main menu.
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
2. In the Deployment dialog box, click the Advanced Options button and specify additional uploading settings in the Advanced Options dialog box that opens:
   - To set the client to the passive mode , select the Passive mode check box. In this mode, the client on your machine connects to the server to inform about being in the passive mode, receives the port number to listen to, and established data connection through the port with the received number. This mode is helpful when your machine is behind a firewall.
   - To have the hidden files and directories (those with names that start with a dot `.`) shown in the Server Browser Tool Window , select the Show Hidden Files check box.
   - Select the Compatibility mode check box to ensure compatibility in child file naming with your FTP server.
     This option is helpful if the remote FTP server reports the following error:

     ```
     Invalid descendant file
                         name <file name>
     ```

     Selecting this option may slow down synchronization with the server.
   - Use the Retrieve accurate files timestamps drop-down list to specify the MDTM FTP command calling policy to retrieve the last-modified time of a given file on the remote host. The available options are:
     - Always - select this option to have MDTM called for every file shown in the Remote Host tool window.
     - On copy - select this option to have MDTM called in the following cases:
       - To check whether a file is up to date when the Overwrite up-to-date files check box in the Options dialog box is cleared.
       - To preserve the actual time stamp of a file during download.
     - Never - select this option to suppress calling MDTM.
   - Select the Limit concurrent connections check box to have PhpStorm restrict the number of connections to be supported simultaneously and specify the maximum number of allowed connections in the text box.
   - In the Control encoding text box, specify the encoding that matches the encoding used by your server. Accept the default value if you are not sure that it supports UTF-8 encoding.
   - Select the Always use LIST command check box to use the standard `LIST` command for listing instead of the `MLSD` command. This lets you avoid problems, for example, failure during upload with the Invalid descendent file name exception if the FTP server supports `MLSD` and returns `cdir` .
   - In the Send keep alive message each text box, specify how often you want PhpStorm to send commands to the server to reset the timeout and thus preserve the connection.
   - From the Use keep alive command drop-down list, choose the commands to be sent to the server to reset the timeout and thus preserve the connection.
   - On some SFTP servers, the SSH banner may be enabled. Every time a connection is established, a pop-up window with an information message may be shown and to continue you would need to click OK.
     To suppress showing the information pop-up window, select the Ignore info messages check box.

Excluding Files and Folders from Upload/Download

On this page:

- Basics
- Excluding a folder on server from upload/download during project creation
- Excluding a folder on server from upload/download after project creation
- Excluding a local folder from upload/download
- Excluding files and folders from upload/download by name
- Removing the exclusion mark

Basics

Suppressing uploading, downloading, and synchronization for files or folders with sources ensures that the sources are protected against accidental update. When applied to non-sources, it saves system resources because huge amounts of media, caches, or temporal files are no longer copied hither and thither.

You may need to suppress upload/download in the following cases:

1. You are going to work with externally created and uploaded source code. To process these remote sources in PhpStorm, you have to download them and arrange them in a project. However, there are some sources that you should not update at all. On the other hand, the folders on the remote host also may contain huge amounts of media, caches, temporal files, that you actually do not need in your work.
2. You have already downloaded the data from the server and arranged them in a PhpStorm project. However, for this or that reason, you need to have some files or folders on the server protected against upload/download, for example, to prevent accidental overwriting.
3. The local copy of an application contains both source code and other data that you do not need to upload. Besides, you want to protect some sources against overwriting by mistake. In this case, you can suppress upload/download for all files and folders that should not be uploaded.

There are two ways to exclude folders from upload/download:

- Explicitly, by marking the corresponding paths as excluded during project creation , or in the Remote Host tool window, or in the Excluded Paths tab of the Deployment dialog box .
  In the Remote Host tool window, you can exclude both entire folders and specific files.
- By name, that is, by specifying patterns that determine the names of files and folders to be excluded in the Exclude Items by Name text box of the Options dialog box.

Separate files can be protected against upload/download only through excluding them by name .

> **Note** In either case, the exclusion is applied recursively. This means that if the path to a folder is explicitly marked as excluded or the folder name matches the pattern, the contents of all its subfolders, if any, are also protected against upload/download.

Excluding a folder on server from upload/download during project creation

1. Initiate creation of a project from existing files by choosing New Project from Existing Files on the main menu. The New Project wizard starts.
2. On the first, Choose Your Scenario , page of the wizard, specify the method to access the remote host, then follow the wizard.
3. On the Choose Remote Path page of the wizard, select the folder and click the icon on the toolbar or choose Project Root on the context menu. PhpStorm marks the selected folder with the icon.
4. Under the project root, specify the folders that you do not need downloaded for some reason. Select the relevant folder and click the Excluded from Download icon or choose Exclude Path on the context menu of the selection.
   Exclusion is applied recursively. This means that if the selected folder contains subfolders they are automatically marked as excluded as well.

Excluding a folder on server from upload/download after project creation

1. Choose the required folder right in the Remote Host tool window:
   1. On the main menu, choose Tools | Deployment | Browse Remote Host or View | Tool Windows | Remote Host .
   2. In the Remote Host tool window that opens, select the relevant server configuration from the drop-down box.
   3. Select the folder to exclude and choose Exclude Path on the context menu of the selection.
2. Add the required folder to the list of excluded paths:
   1. Open the Deployment dialog box by doing one of the following:
      - Choose Tools | Deployment | Configuration on the main menu.
      - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
   2. In the Deployment dialog box, switch to the Excluded Paths tab. The tab shows list of previously excluded local and remote folders.
   3. Click the Add deployment path button. An empty line is added to the list.
   4. Click the Browse button . The Select remote excluded path dialog box that opens shows the data on the host accessed through the selected server configuration. Select the required folder.
   5. When you OK, you return to the Excluded Paths tab, where the selected remote folder is added to the list.

Excluding a local folder from upload/download

1. Open the Deployment dialog box by doing one of the following:
   - Choose Tools | Deployment | Configuration on the main menu.
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Deployment under Build, Execution, Deployment.
2. In the Deployment dialog box, switch to the Excluded Paths tab. The tab shows list of previously excluded local and remote folders.
3. Click the Add local path button. In the empty line that is added to the list, specify the location of the folder to be protected against upload/download. Type the path manually or click the Browse button and choose the required folder in the dialog that opens .

Excluding files and folders from upload/download by name

1. Open the Options dialog box by doing one of the following:

- On the main menu, choose Tools | Deployment | Options.
- Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Build, Execution, Deployment node, and then click Options under Deployment.

2. In the Options dialog box that opens, specify the patterns that define the names of these files and folders in the Exclude items by name text box. Use semicolons as delimiters. Wildcards are welcome.
   The exclusion is applied recursively. This means that if a matching folder has subfolders, the contents of these subfolders are not deployed either.

Removing the exclusion mark

To return a file or folder to upload/download, select it and choose Remove Path from Excluded on the context menu of the selection. Returning a folder to upload/download automatically affects all its subfolders and files.

Uploading and Downloading Files

On this page:

- Basics
- Uploading files and folders manually
- Uploading application sources automatically before application start
- Uploading changed files automatically
- Uploading checked-in files immediately after commit
- Suppressing automatic upload
- Downloading files

Basics

There are the following main ways to synchronize project files with their copies on the local or remote server:

- Manually , through a menu command.
- Automatically, every time a file is updated . In this case, the default server configuration is used.
  You can additionally configure this upload method to have only manually saved files uploaded.
- Automatically, before launching the application or starting a debugging session. This method is available for PHP applications only,
- Automatically after the changes to the files are checked in to your version control repository.

Besides uploading files, you can also have PhpStorm download files that were deployed according to the default server configuration .

Uploading files and folders manually

1. Choose Tools | Deployment on the main menu.
2. Do one of the following:
   - To run the upload according to the default configuration, choose Upload to <default server configuration>.
   - To run the upload according to a specific configuration, choose Upload to. Then from the pop-up menu, select the server configuration that defines access to the required server.

Uploading application sources automatically before application start

When working on a PHP application, you can have its sources automatically uploaded to a server every time you run the application or initiate a debugging session from PhpStorm.

1. During creation of a run/debug configuration , click the Add button + and choose the Upload Files to Remote Host task from the list.
2. In the Upload to Remote Host dialog box that opens, specify the following:
   - The server configuration to access the required server.
   - The local root folders to upload. All the folders and files under the specified roots will be uploaded.

Uploading changed files automatically

PhpStorm considers a local file changed as soon as it is saved either automatically or manually. PhpStorm performs automatic save on a regular scheduled basis. Manual save is invoked by choosing File | Save All on the main menu or pressing `Ctrl+S`.

1. Open the Options dialog box by doing one of the following:
   - On the main menu, choose Tools | Deployment | Options.
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Build, Execution, Deployment node, and then click Options under Deployment.
2. In the Upload changed files automatically to the default server drop-down list, specify when you want PhpStorm to upload changed files:
   - To have PhpStorm upload any saved file no matter whether the save was invoked manually or automatically, choose Always.
   - To have PhpStorm upload only files that were saved manually, choose On explicit save action.

Uploading checked-in files immediately after commit

1. Start checking in your changes .
2. In the After Commit area, choose the target server from the Upload file to drop-down list. Choose one of the existing server access configurations or create a new one: click the Browse button ... and configure access to the relevant server in the Deployment dialog box that opens.
3. To have PhpStorm automatically upload checked in files to the chosen server, select the Always use selected server check box.

Suppressing automatic upload

1. Open the Options dialog box by doing one of the following:
   - On the main menu, choose Tools | Deployment | Options.
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Build, Execution, Deployment node, and then click Options under Deployment.
2. From the Upload changed files automatically to the default server drop-down list, choose Never.

Downloading files

Choose Tools | Deployment | Download from <default server configuration> on the main menu. Note that this functionality is available only if you have previously appointed a configuration as default.

Accessing Files on Web Servers

On this page:

- Basics
- Accessing a server
- Creating a folder on a server
- Removing a file or folder on a server
- Renaming a file or folder on a server
- Moving a file or folder

Basics

Once you have set up synchronization between your local application sources and the application sources on a server, you can create new folders, move, rename, and delete existing files and folders. You can also compare files and folders on the server with their local versions.

Please note, that for the sake of simplicity any file or folder under the project root is referred to as local and any file or folder on the server is referred to as remote, even if the server is installed on your machine. For details, see Working with Web Servers: Copying Files .

Although PhpStorm supports direct editing of individual non-project files on servers, to get full coding assistance either download the sources from the server and arrange them in a PhpStorm project or upload the local sources to the server. In either case, to keep your local and remote sources synchronized, configure automatic upload using the Upload changed files automatically to the default server drop-down list in the Options dialog box.

Access to servers is controlled through server configurations of the type FTP, FTPS, SFTP, or Local or Mounted Folder.

Accessing a server

1. Open the Remote Host tool window. On the main menu, choose one of the following:
   - View | Tool Windows | Remote Host
   - Tools | Deployment | Browse Remote Host
2. Select the required server configuration from the drop-down list. The tool window shows a tree view of files and folders under the server configuration root. If no relevant server configuration is available in the drop-down list, click the Browse button and configure access to the server in the Deployment dialog box that opens.
   For details, see Configuring Synchronization with a Web Server .

Creating a folder on a server

Select the parent directory and choose Create Folder on the context menu.

Removing a file or folder on a server

Select the required file or folder and choose Delete on the context menu.

Renaming a file or folder on a server

Select the required file or folder in the tree view, choose Rename on the context menu, and specify the new name in the Rename dialog box that opens.

Moving a file or folder

1. Select the file or folder and choose Cut on the context menu.
2. Select the folder to move the file or folder to and choose Paste on the context menu. Then confirm the changes in the Move remote Items dialog box that opens.

   Alternatively, select the required file or folder and drag it to the new location. Then confirm the changes in the Move remote Items dialog box that opens.

Comparing Deployed Files and Folders with Their Local Versions

On this page:

- Basics
- Opening the Remote Host tool window
- Comparing a remote folder with its local version
- Comparing a remote file with its local version
- Comparing a local folder with its remote version
- Comparing a local file with its version on a server
- Comparing two folders in the Difference Viewer
- Synchronizing contents of folders

Basics

Correspondence between files and folders under the project root and their copies on a server is set through server access configuration mappings . Please note, that for the sake of simplicity any file or folder under the project root is referred to as local and any file or folder on the server is referred to as remote, even if the server is installed on your machine. For details, see Working with Web Servers: Copying Files . For details, see Configuring Synchronization with a Web Server .

It is clear, that a local file or folder can be mapped to an unlimited number of remote counterparts, while each remote file or folder is mapped to one and only one file or folder respectively.

Consequently, for each remote file or folder, PhpStorm unambiguously detects its local version to compare with, so you can have remote items compared with their local counterparts at any time. On the contrary, remote versions of local files or folders cannot be identified uniquely, therefore PhpStorm can compare local items only with their remote versions mapped through the default server access configuration .

PhpStorm provides a dedicated Differences Viewer for Folders for comparing files in remote folders and their local versions against the file size, content, or timestamp. The Differences Viewer shows the contents of the local and remote directories in the Item List, in the left and right panes respectively. The contents of the selected file are shown in the lower pane, with the differences being color-highlighted.

The remote files in the Difference Viewer have the status `read-only` . This means that you cannot update them directly in the Difference Viewer. Make all the necessary changes to the local version of the file in question and upload the updated file to the server.

Besides exploring differences, the tool also provides interface for synchronizing the contents of folders.

Opening the Remote Host tool window

Do one of the following:

- On the main menu, choose Tools | Deployment | Browse Remote Host.
- On the main menu, choose View | Tool Windows | Remote Host. Note that the tool window can be accessed this way after you have opened it using the Tools | Deployment | Browse Remote Host.

Comparing a remote folder with its local version

1. Open the Remote Host tool window .
2. Select the folder in question. Then choose Tools | Deployment | Sync with Local on the main menu or Sync with local on the context menu of the selection.
3. In the Differences Viewer for Folders that opens, explore the differences and apply them, if necessary, using the » and « buttons. For details, see Viewing Differences Between Files .

Comparing a remote file with its local version

1. Open the Remote Host tool window .
2. Select the file in question, and then choose Compare with local version on the context menu of the selection.
3. In the Differences Viewer for Files dialog box, that opens, explore the differences and apply them, if necessary, using the » and « buttons. For details, see Viewing Differences Between Files .

Comparing a local folder with its remote version

This action is available only if you have a default server access configuration appointed.

1. Select the folder in question in the Project tool window, and then choose Tools | Deployment | Sync with Deployed to <default server access configuration> on the main menu.
2. In the Differences Viewer for Folders that opens, explore the differences and apply them, if necessary, using the » and « buttons. For details, see Viewing Differences Between Files .

Comparing a local file with its version on a server

This action is available only if you have a default server access configuration appointed.

1. Select the file in question in the Project tool window, and then choose Tools | Deployment | Compare with Deployed Version on <default server access configuration> on the main menu.

   Note When no default server access configuration is appointed, this menu item is disabled and its name is changed to Compare Local File with Deployed Version.

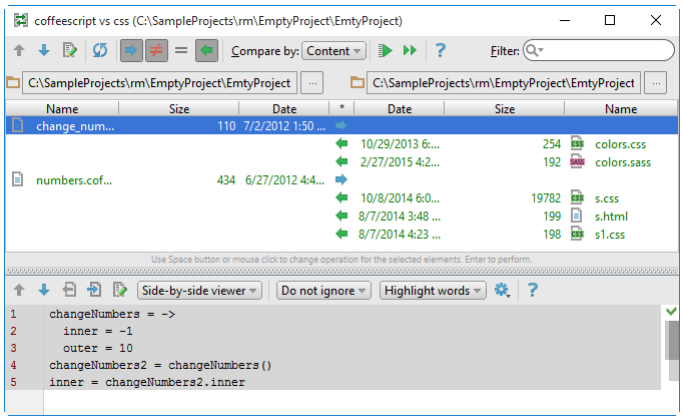
2. In the Differences Viewer for Files dialog box, that opens, explore the differences and apply them, if necessary, using the » and « buttons. For details, see Viewing Differences Between Files .

Comparing two folders in the Difference Viewer

Configure the layout of the Items List. Use the toolbar buttons to narrow down or widen the set of items to show. For example, show or hide files that are present only locally or remotely, equal files, different files, files excluded from synchronization , etc.

Specify the parameter for comparison. In the Compare by drop-down list, select one of the possible options (contents, size, or time stamp). Filter the folders' contents. To do that, type filtering string in the Filter text field, and press `Enter` to apply it. Using the asterisk `*` wildcard to represent any number of characters is welcome. To switch to another pair of folders to compare, update the fully qualified paths to them. Click the Browse button ... next to the Paths read-only fields and choose the required folders in

the dialog box that opens . Explore the detected differences between files in the Differences Pane.



Synchronizing contents of folders

The contents of the remote folder are always shown in the right-hand pane of the Items List, the contents of its local version are always shown in the left-hand pane.

1. For each pair of items, in the * field specifies the action to apply. Click the icon in the field until the required action is set.
   - the file will be uploaded, possibly overwriting the remote version.
   - the file will be downloaded, possibly overwriting the local version.
   - the files are treated identical with regard to the selected criterion of comparison. No action will be performed by default.
   - the files differ with regard to the selected criterion of comparison. No action will be performed by default. Explore the differences in the Differences Pane of the Difference Viewer and change the intended action by clicking the icon.
     The remote files in the Difference Viewer have the status `read-only`. This means that you cannot update them directly in the Difference Viewer. Make all the necessary changes to the local version of the file in question and upload the updated file to the server.
   - the file is present only locally or remotely and will be removed.
2. Do one of the following:
   - To synchronize the currently selected item, click the Synchronize Selected button on the toolbar.
   - To synchronize all the items, click the Synchronize All button on the toolbar.

Editing Individual Files on Remote Hosts

On this page:

- Introduction
- Editing files on remote hosts

Introduction

Once you've set up synchronization with a remote host, you can open individual files directly from the remote host and edit them in PhpStorm, without adding/downloading them to the local project. Note that for such files not included in a project, some PhpStorm features are not supported.

To take advantage of debugging, refactorings, and other advanced features, consider including the files into a project; see Accessing Files on Web Servers for details.

Code completion is limited to standard PHP library methods and the methods used in the edited file.

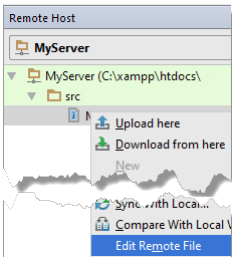
Editing files on remote hosts

To edit a file on a remote host

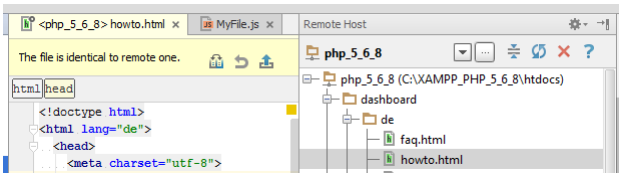
1. Open the Remote Host tool window. On the main menu, choose one of the following:
   - View | Tool Windows | Remote Host
   - Tools | Deployment | Browse Remote Host

2. Select the required server configuration from the drop-down list. The tool window shows a tree view of files and folders under the server configuration root. If no relevant server configuration is available in the drop-down list, click the Browse button and configure access to the server in the Deployment dialog box that opens.
   For details, see Configuring Synchronization with a Web Server .

3. To start editing a file, double click its name or select the file name in the Remote Host Tool Window , and choose Edit Remote File on the context menu:

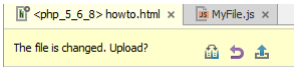


   The file opens in the PhpStorm editor, without being added or downloaded to the local project.



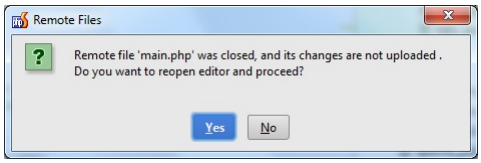
   When you work with a remote file, a special toolbar appears at the top of the editor, showing the editing status ("The file is identical to the remote one" or "The file has been changed. Upload?".

   Remote files can be easily distinguished from local ones by looking at the annotation, which includes the server name (in our case it is <php_5_6_8>) .



4. When you are done editing the file, do one of the following:
   - To upload the file to the remote host, click .
   - To compare the currently opened file with the last uploaded version, click . The files are opened in the standard PhpStorm Difference Viewer, see Differences viewer for files .
   - To discard the changes introduced to the file after the last upload, click .

As it has been mentioned above, an individual file is not added to a project. As a result, all the changes to it are discarded as soon as you close the file or the currently opened project unless these changes have been uploaded. To prevent losing your data accidentally, PhpStorm displays the following dialog box when you attempt to close the edited file or the entire project:



Using Local History

This section describes how to use Local History , which is your personal real-time version control system. Local History is independent of external version control systems and works with the directories of your project even when they are not under any VCS control.

Local history is cleared when you install a new version of PhpStorm or when you invalidate caches . Therefore, check in the changes to your version control system before performing these operations.

This section describes how to:

- View local history of a file or folder
- View local history of a selection
- View recent changes
- Restore files from local history
- Mark local versions with labels

Putting Labels

On this page:

- Basics
- Adding a label to a local version

Basics

Before embarking on a risky change to your source code, it is a good idea to mark the stable version with some meaningful label. This will help you quickly roll back to a safe version.

Labels apply to a whole project.

Adding a label to a local version

To add a label to a local version

1. Select a file or folder in the Project tool window, or open a file in the editor.
2. Do one of the following:
   - On the main VCS menu, or on the context menu of the selection, choose Local History | Put Label.
   - Press `Alt+Back Quote` and choose Put Label command from the VCS Operations quick list.
3. In the Put Label dialog box, type the label name.

Restoring a File from Local History

Rolling back changes from the local history works same way as in the regular version control.

To roll back changes in the local history

1. Open the Local History view .
2. Select the version you want to roll back to.
3. On the context menu of the selection, choose Revert.

Viewing Local History of Source Code

On this page:

- Basics
- Viewing local history of a source code block

Basics

Besides file history , you can track local changes for a selected block of source code . his history shows only those changes that affect the code fragment.

Viewing local history of a source code block

To view local history of a source code block

1. In the editor, select a fragment of source code.
2. Do one of the following:
   - On the main VCS menu, or on the context menu of the selection, choose Local History | Show History for Selection.
   - Press `Alt+Back Quote`, and choose the desired command from the VCS Operations quick list.

Viewing Local History of a File or Folder

On this page:

- Basics
- Viewing local history

Basics

Local History makes it possible to view changes made to a certain file or a whole directory. Each entry in the Local History dialog box is displayed with its time stamp, action and optional label. So doing, the local history for a file includes all changes that affect both the selected file and the whole project; local history for a folder shows changes to the source code tree in general. You can explore changes, selecting the respective row in the Local History dialog box.

Viewing local history

To view local history

1. Select a folder or file in the Project tool window, or open a file in the editor.
2. Do one of the following:
   - On the main VCS menu, or on the context menu of the selection, choose Local History | Show History.
   - Press `Alt+Back Quote`, and choose the desired command from the VCS Operations quick list.
   - Use View Recent Changes that shows a summary of recent changes in a single pop-up list. Clicking an entry in this list shows the respective Local History.
3. Use the Local History view to compare local versions and revert changes.

Viewing Recent Changes

On this page:

– Basics

– Viewing recent changes

Basics

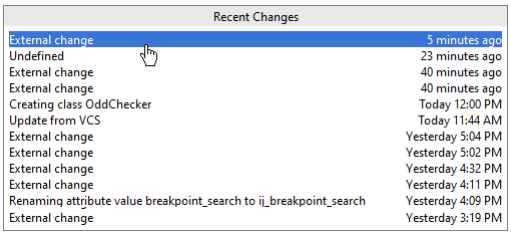
PhpStorm enables you to view the summary of the recent changes to a project. From the Recent Changes pop-up window, you can browse through the history of changes, navigate to a particular change, compare versions, and roll changes back.

Viewing recent changes

To view recent changes

1. On the main menu, choose View | Recent Changes, or press `Shift+Alt+C`.

2. In the Recent Changes pop-up window, select the change you are interested in:



3. In the dialog box that opens, review differences and discard changes, if necessary.

Version Control with PhpStorm

PhpStorm provides integration with several Version Control Systems (referred to as VCS in documentation). This includes both support of features specific for each VCS as well as unified interface and management for common VCS tasks.

In this part:

- Concepts of Version Control
- Accessing VCS Operations
- Enabling Version Control
- Configuring Version Control Options
- Common Version Control Procedures
- VCS-Specific Procedures

Concepts of Version Control

In this part you can find descriptions of procedures that are common to all supported version control systems, or specific to certain VCS integrations:

- Supported Version Control Systems
- Directory-Based Versioning Model
- Unified Version Control Functionality
- Changelist
- Local, Repository, and Incoming Changes
- Local History
- Patches
- Shelved Changes

Supported Version Control Systems

The list of supported integrations of version control systems is determined by the set of currently enabled plugins .

The basic principles of getting started with the supported version control systems (VCS) in PhpStorm are rather similar, though some commands and settings are specific and conform to the version control system conventions. In addition to the common information, you can find VCS-specific procedures in the following sections:

- Using Git Integration
- Using GitHub Integration
- Using Subversion Integration
- Using Mercurial Integration
- Using CVS Integration
- Using Perforce Integration

Directory-Based Versioning Model

PhpStorm adopts a directory based model for version control association. A version control system is assigned to a project directory and/or additional directories that are part of or related to the project. Directories under version control are not required to be located under the project root. They can reside in any accessible location.

All the settings files in the `.idea` directory should be put under version control except the `workspace.xml`, which stores your local preferences. The `workspace.xml` file should be marked as ignored by VCS .

Unified Version Control Functionality

In addition to support for general and individual VCS commands, PhpStorm provides several unique features that simplify and speed up the work with any version control system.

- For the projects with VCS support enabled, the standard VCS actions (commit, update, revert, show differences and show history) are added to the main toolbar.
- Commit and update an entire project.
- Uniform interface for configuring common version control system settings.
- Changelists support for all integrated version control systems.
- Next, Previous, Rollback, Show Difference actions are available from the dedicated gutter bar in changed locations.
- View revision history for file/directory.
- Automatic checkout of all affected files when refactoring.
- Advanced Version Control tool window, with multiple dedicated tabs: History, Status, Update Info, etc.

Mind the difference in terminology in the different version control systems. For example, to denote the check-in functionality, Git uses the term *commit*, Subversion uses *submit*, etc.

Changelist

A changelist is a set of changes in files that represents a logical change in source code. The changes specified in a changelist are not stored in the repository until committed (pushed).

Any changes made to the source files, are automatically included in the active changelist. Initially, the Default changelist is active, but you can make any other changelist active. The active changelist is displayed on top of the Version Control tool window, with the name being highlighted in bold font.

In addition to the Default changelist, you can create new changelists, delete existing ones (except for the Default changelist), and move files between changelists.

All modified, deleted, unversioned and other files are managed in the Version Control tool window . From this window you can:

- Commit (push) changelists.
- Create new changelists (if you want to keep an eye on certain files and changes).
- Remove existing changelists and set the default changelists.
- Rollback modified files in changelists.
- Add the unversioned files and directories to the version control.
- Move files between changelists.
- Show differences on selected files.
- Refresh the list of VCS changes.
- Jump to the source code from within a changelist.
- Shelve (stash) and unshelve (unstash) changes.

Local, Repository, and Incoming Changes

PhpStorm distinguishes among three categories of changes: Local, Repository, and Incoming.

Local changes

The changes that you have introduced to your local working copy but have not yet checked in to the repository.

Repository changes

The changes that you and other users have checked in to the repository.

Incoming changes

The changes that have been checked in to the repository but that you have not yet checked out locally.

Warning! Repository and incoming changes are not supported by Git and Mercurial integrations.

All these types of changes are displayed in the respective tabs of the Version Control tool window. Information about changes is stored in the history cache. Configure the size of this cache and frequency of refreshing information in the General VCS Settings .

Local History

Your source code constantly changes as you edit, test, or compile. Any version control system tracks the differences between the committed versions, but the local changes between commits pass unnoticed. Local History is your personal version control system that tracks changes to your source code on your computer and enables you to compare versions and roll changes back, if necessary. Local History is always at your disposal, no steps are required to enable it.

Local History is independent of external version control systems and works with the directories of your project even when they are not under any VCS control. It applies to any structural artifacts: a project, a directory or package, a file, a class, class members, tags, or selected fragment of text.

Unlike usual version control systems, Local History is intended for your personal use, it does not support shared access.

With Local History, PhpStorm automatically tracks changes you make to the source code, results of refactoring, and state of the source code based on a set of predefined events (testing, deployment, commit or update).

Local History revisions are marked with labels, which are similar to versions in traditional version control systems. Labels based on predefined events are added to the local revisions automatically; besides that, you can put your own labels to the project artifacts to mark your changes. Reverting or viewing differences are performed against these labels.

Local History has certain limitations:

- Tracking local changes is only possible for textual files. Binary files do not have Local History.
- For files larger than 1 Mbyte, Local History tracks only the very fact of changes, but does not preserve the respective contents.
- Local History does not support shared access.

Patches

PhpStorm helps you create and apply patches to the source code. A patch is a file in the standard text format that has the *.patch extension, and contains a list of differences between two sets of source files.

Patches only contain changes to textual files. Changes to binary files cannot be patched.

This concept is tightly related to the concept of Shelved Changes .

Shelved Changes

You can run into a situation when you are short of time to bring your source code to a certain required condition or you need to work on an urgent high priority task. In this case you might want to put some changes aside and continue working on a stable version.

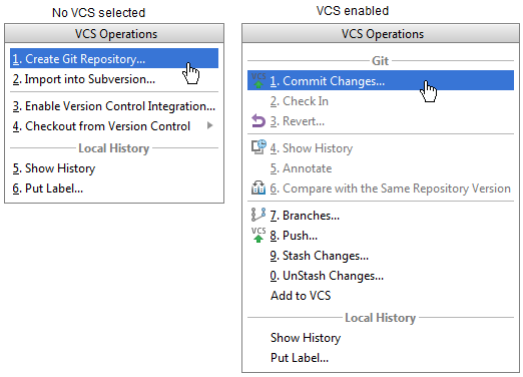
With PhpStorm, you can use shelves for storing postponed changes temporarily. In due time, the desired changes can be taken back from the shelf (unshelved).

PhpStorm enables shelving both separate files and entire changelists. Accordingly, you can unshelve entire shelves or specific files.

Accessing VCS Operations

Use the VCS Operations Pop-up to quickly invoke most required commands.

Note that the composition of VCS commands available on the pop-up menu, depends on the specific VCS.



To quickly invoke a VCS command using VCS Operations Pop-up

1. Open VCS Operations pop-up, in one of the following ways:
   - On the main menu, choose VCS | VCS Operations Pop-up.
   - Press `Alt+Back Quote`.

2. Choose command from the VCS Operations list. To do that, perform one of the following actions:
   - Click the desired command in the list.
   - Use up and down arrow keys to select the desired command, and then press `Enter`.
   - Press the number key that corresponds to the desired command in the list.

Enabling Version Control

PhpStorm supports version control integration at two levels:

- At the IDE level, VCS integration is provided through a set of bundled plugins enabled by default.
- At the project level, VCS integration is enabled by associating project folders with one or several version control systems.

This section describes how to:

- Associate a project root with version control system.
- Associate a directory with version control system.
- Change the associated VCS for the project root or directory.

Associating a Project Root with a Version Control System

PhpStorm allows you to quickly enable your project's integration with a version control system, and associate it with the project root. For instructions on how to associate separate project directories with different version control systems, refer to Associating a Directory with a Specific Version Control System .

To assign a version control system to the project root

1. Choose VCS | Enable Version Control Integration on the main menu, or press `Alt+Back Quote`, and select Enable Version Control Integration....
2. In the Enable Version Control Integration dialog box that opens, select a version control system from the drop-down list that you want to associate with your project root .

Associating a Directory with a Specific Version Control System

PhpStorm supports a directory-based versioning model , which means that each project directory can be associated with a different version control system.

Associating a directory with a version control system

To associate a directory with a version control system, follow these steps:

1. Open version control settings (File | Settings | Version Control). This page shows a list of project directories and version control systems associated with them (if no directories have been added, the list only contains the project root).
2. Click the Add button + on the right.
3. In the Add VCS Directory Mapping dialog box that opens, select the Directory option. Type the path to the directory that you want to associate with a version control system, or click the Browse button and select the directory in the dialog that opens .
4. From the VCS drop-down list, select the version control system that will be used to control the files in this directory. Note that this list only contains the version control systems for which the corresponding Plugins are currently enabled (see Managing Plugins for details).
5. Optionally, click the Configure VCS button that allows you to specify the settings for the selected version control system. The same settings are also available under the Version Control settings node.
6. Click OK to save the mapping and return to the Version Control page.

Managing unregistered directories

For projects with Git or Mercurial integration enabled, PhpStorm scans project directories to check if there are Git/Mercurial repositories that are not controlled by the IDE. If such repositories are detected, PhpStorm displays a notification.

To add an unregistered root, click the Add roots link in the notification. Alternatively, open the Version Control settings page , select the unregistered root you want to add (they are marked grey) and follow the procedure Associating a Directory with a Specific Version Control System .

If you do not want to be notified about these roots again, click the Ignore link in the notification. Note that if new unregistered repositories are added to the project, PhpStorm will notify you about them.

To change the version control system associated with a directory

1. Open the Version Control settings page. This settings page displays a table of directories with associated version control systems.

2. In the table, locate the row that corresponds to the directory which you want to put under another version control system.

3. Click the VCS column. From the drop-down list that appears, select a new version control system.

   Tip Optionally, click the Configure VCS button. The Version Control Configurations dialog box opens where you can configure settings for the selected version control system (see the corresponding configuration settings in the Version Control settings page reference for details).

4. Click OK to save the new mapping and close the Version Control dialog box.

Configuring Version Control Options

Configuring version control options involves:

- Configuring General VCS Settings
- Configuring VCS-Specific Settings

Configuring General VCS Settings

General version control settings apply to all version control systems integrated with PhpStorm. General settings are specified on the Version Control page of the Settings dialog box, and include defining actions that require confirmation, background operations, ignored files, issue navigation, and depth of history.

To configure general version control settings, follow these general steps

1. Open the Settings dialog box and then click Version Control.
2. Specify which version control related actions should require confirmation .
3. Specify which operations should be performed in the background .
4. Create a list of files to be ignored by version control systems.
5. Configure history cache handling .
6. Define issue navigation rules to switch from check-in comments to corresponding issues in a bug tracking system.

Specifying Actions to Confirm

You can define that certain version control related activities should be performed only upon confirmation from your side. The activities for which confirmation is required are specified in the Confirmation settings page.

To specify which actions should require confirmation

1. Below the Version Control node in the settings, select the Confirmation page.

2. In the Files Creation/Deletion area, define how files created or deleted in PhpStorm should be added to or removed from a version control system. The available options are:
   - Show options before adding to version control
   - Add silently
   - Do not add

3. In the Display options dialog when these commands are invoked area, specify the commands, for which you want PhpStorm to display the Options dialog box. The available options are:
   - Check Out
   - Status
   - Get Latest Version
   - Update
   - Undo Check Out

4. Configure additional settings for working with changelists by selecting or clearing the corresponding check boxes:
   - Specify whether the read-only state of files should require explicit cancellation.
   - Specify whether meaningful comments on committing files to the repository should be required.
   - Specify whether uncommitted changes should be moved to another changelist.
   - Specify whether and how to create a changelist if the commit operation fails.

Specifying Actions to Run in the Background

You can enable background execution of certain version control related activities. These activities are specified in the Background settings page.

To specify the operations to run in the background

1. Below the Version Control node in the settings, select the Background page.

2. Enable background execution of the necessary actions by selecting the corresponding check boxes. Background execution can be set for the following actions:
   - Update
   - Commit
   - Checkout
   - Edit/Checkout
   - Add / Remove
   - Revert
   - History Cache Handling
   - Detecting "changed on server" conflicts

Configuring Ignored Files

On this page:

- Basics
- Defining a list of ignored files
- Patterns

Basics

Sometimes you may need to leave files of certain types unversioned. These can be VCS administration files, artifacts of utilities, backup copies, etc. You can create a global ignore list that will be stored in the workspace file and applied to all supported version control systems.

The files you want to ignore can be appointed explicitly by their names of through name patterns with wildcards. To ignore a directory, you need to specify the full path to it relative to the project root.

Use the Ignored Files settings page to list files that must be excluded from version control operations.

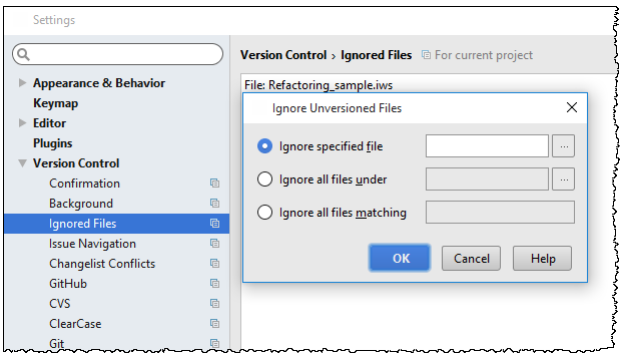
> **Tip** If the version control system that you are using has its own ignore facilities, use the corresponding native command provided by the version control integration.

Note that once you've added a file to a version control system, ignoring it will have no effect. You need to remove it from your VCS first.

Defining a list of ignored files

1. Open the Ignored Files settings page by doing one of the following:
   - Under the Version Control node of the Settings dialog box, click Ignored Files.
   - In the Local Changes tab of the Version Control tool window, click the Configure Ignored Files toolbar button .

   The Ignored Files dialog box opens.



2. Click + (`Alt+Insert`) to create a new entry, or select an existing entry and click (`Enter`). The Ignore Unversioned Files dialog box opens.
3. Explicitly specify the files/directories to be ignored or define file name patterns. Do one of the following:
   - Choose the Ignore specified file option and specify the file name relative to the project root, for example, `my_folder/my_subfolder1/my_subfolder2/my_file` . Type the name manually or click the Browse button and select the desired file in the Select File to Ignore dialog box.
   - Choose the Ignore all files under option and specify the directory whose contents should be ignored. Type the directory name relative to the project root, for example, `my_folder/my_subfolder1/` , or click Browse button and select the desired folder in the Select Directory to Ignore dialog box.
     The rule is applied recursively to all the subdirectories of the specified directory. If a directory has several subdirectories and you want only one of them ignored, specify the required directory explicitly, for example, `my_folder/my_subfolder1/my_subfolder2/` .
   - Select the Ignore all files matching option and type the pattern that defines the names of files to be ignored. The rule is applied to all the directories under the project root. Using wildcards in combination with slashes (`/`) to restrict the scope to a certain directory is not supported.
     Patterns that define files to ignore, make use of two wildcards .
4. Create as many entries as you need and close the dialog box.

> **Tip** You can also add files to the ignore list on-the-fly by using the Ignore command on the context menu of a newly added file under the Unversioned Files node in the Local Changes tab of the Version Control tool window .

Patterns

Two characters are used as the wildcards:

- `*` : this character matches anything.
- `?` : this character matches a single character.

For example, `*.iml` will ignore all the files with the extension `iml` ; `*.?ml` will ignore all the files with the extension ending with `ml` .

Configuring History Cache Handling

You can configure handling of the history cache in the Background settings page.

To configure history cache handling

1. Below the Version Control node in the settings, select the Background page.

2. Set the cache scope. Depending on the version control system you are using, do one of the following:
   - Specify the maximum number of changelists to be stored in the cache.
   - Specify the maximum number of days for a changelist to be stored in the cache.

3. Specify whether and how often (in minutes) you want your version control system to refresh the cache.

Configuring VCS-Specific Settings

Certain settings are applicable to the version control systems assigned to the whole project, or its directories . The others are related to the selected version control systems. Use the respective VCS pages under the VCSs node of the Settings dialog box to define VCS-specific settings.

To configure VCS-specific settings, follow these general steps

1. Open the Settings dialog box and then click Version Control.
2. Click a page that corresponds to the VCS to be configured.
3. Set up options as required. For detailed information, see VCS-specific pages of the Version Control settings.

Common Version Control Procedures

PhpStorm makes it possible to interact with the different version control systems (VCS). Regardless of which VCS you use, certain basic operations are common to all (or almost all) of them. This section covers these basic operations and explains how to perform them from within PhpStorm:

- Adding Files to Version Control
- Browsing Contents of the Repository
- Getting Local Working Copy of the Repository
- Changing Read-Only Status of Files
- Checking in Files
- Checking Project Files Status
- Copying, Renaming and Moving Files
- Deleting Files from the Repository
- Handling Differences
- Handling Issues
- Managing Changelists
- Refreshing Status
- Reverting Local Changes
- Reverting to a Previous Version
- Shelving and Unshelving Changes
- Updating Local Information
- Using Patches
- Viewing Changes Information
- Accessing the Authentication to Server Dialog

Adding Files to Version Control

If a new file is created with PhpStorm in a directory that is already under version control, it automatically adds to the active changelist with the status Added. All you have to do, is to commit this change.

PhpStorm's behavior on adding files is configurable in the General Settings tab of the Version Control dialog.

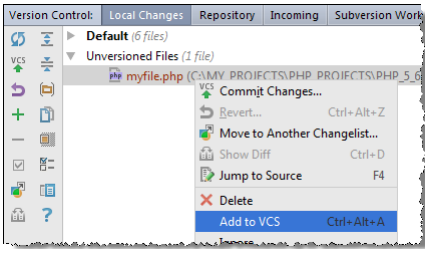
If a new file is created outside of PhpStorm, or silent adding is disabled, you have to explicitly add it to the version control.

Another approach is VCS-specific. You can import an entire directory to your repository, provided that you have the corresponding access rights. This action is helpful for putting a whole project under version control.

To explicitly add a file to version control

1. Select file in the Project tool window.

2. On the main Version Control menu or on the context menu of the selected file, choose <VCS> | Add.
   Alternatively, use the Version Control tool window. Select the desired files in the Unversioned files changelist in the Local Changes tab, and choose Add to VCS on the context menu, or just drag it to the target changelist.



3. Select the added file in the changelist and check in (commit) changes .

Browsing Contents of the Repository

Prior to checking files out, you can browse the contents of the repository. This action is available for CVS, Git and SVN integrations.

- Browsing CVS Repository
- Browsing Subversion Repository

Browsing contents of CVS, Git and SVN repositories is always available, even when the respective VCS is not enabled in project. All you need is a valid user account.

Getting Local Working Copy of the Repository

On this page:

- Basics
- Checking out

Basics

As soon as version control support is enabled in PhpStorm, you can retrieve the data from the repository. Depending on your purpose and workflow, choose one of the following approaches:

- Check out the repository sources to a content root of an existing project . Then set up version control in the project, if it has not been done before, and put the downloaded sources under control of the VCS used.
- Check out the repository sources to a location of your choice and have PhpStorm create a project around them . This does not require any extra steps from your side. PhpStorm arranges the downloaded sources in a project itself and suggests to open it as soon as the check-out is completed.
  Upon your consent, PhpStorm opens the newly created project, where you only need to set up version control.

> Warning! This approach is not available for Perforce integration.

Checking out

The check-out procedure depends on the type of VCS you use. Refer to the following sections for details:

- Retrieving Files from a CVS Repository
- Retrieving Files from an SVN Repository
- Cloning a Remote Git Repository
- Cloning a Remote Mercurial Repository
- Using Perforce Integration

Changing Read-Only Status of Files

In this section:

- Basics
- Enabling explicit removal of read-only status
- Changing writable status by icon
- Example

Basics

Different version control systems have different semantics for the action of removing read-only status from a file so that you can edit it. Some systems never put read-only status on local files at all unless specifically configured to do so (i.e. the system is configured to support the file locking model).

Different version control systems use different names for this action: check out, edit, Open for Edit, or Get. Regardless of the terminology used by your VCS, if it sets read-only status on your local working files, you can remove read-only status and make files writeable from within PhpStorm, which will also take care of setting a lock on the server, or take whatever other action is required by the VCS, via the respective VCS integration.

This behavior is configurable in the Confirmation page of the Version Control settings.

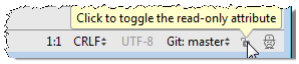
Enabling explicit removal of read-only status

To enable explicit removal of read-only status:

- In the Confirmation page of the Version Control settings , check the option Show "Clear Read-Only Status" Dialog.

Changing writable status by icon

You can make a file writable using the lock icon in the Status bar. Open the desired file in the editor, and click the lock, as shown below:



Example

Removing read-only status in Perforce looks as follows:

1. With the Show "Clear Read-Only Status" Dialog option enabled, an attempt to edit a file brings up the respective dialog box.
   If you click the radio button Using file system, the file will not be added to the default changelist. If you click radio button Using version control integration, the file is added to the default changelist.

Checking In Files

In this section:

– Basics

– Checking changed files in

Basics

Different version control systems have different semantics for the action of uploading changed files to the repository. Two common terms are check in and commit.

In those version control systems, for example, Git , that distinguish between local and remote repositories, the term commit denotes uploading changes to a local repository. Uploading changes to a remote repository is referred to as push.

Regardless of the terminology, you can perform this operation with the VCS configured for a directory from within PhpStorm.

Checking changed files in

To check in (commit) changed files, perform these general steps

1. In the Version Control tool window, select one or more files you want to check in (commit) to version control.

2. Open the Commit Changes dialog box by doing one of the following:
   – On the Version Control tool window toolbar, or on the main toolbar, click VCS.
   – Press `Ctrl+K`.
   – On the main menu, choose VCS | Commit Changes.

3. Review the changes to be committed in the Details pane. To do that, unfold the Details pane if it is hidden, and select the file in question in the Changed Files area.
   The Details pane shows the base version and the local copy of the selected file.

   Examine the details of each change:
   – To move to the next updated piece of code, click the Next Change button .
   – To return to the previous updated code fragment, click the Previous Change button .
   – To expand or narrow the context of an updated code fragment, position the cursor at the change in question, click the More/Less Lines button , and then specify the number of lines to be shown above and below the current code fragment.

4. Add a commit comment. As you type, PhpStorm checks the spelling and highlights words in question, provided that the Spelling code inspection is enabled.

5. Specify which actions should be performed on the files before and after submitting them to the repository.

6. Click the Submit/ Commit button to launch the Check-in Changes operations.

   Tip For Git and Mercurial. To have the changes immediately pushed to your Git or Mercurial repository, do one of the following:
   – Hover the mouse pointer over the Submit/ Commit button and select Commit and Push on the context menu.
   – From the Submit/ Commit drop-down list, select Commit and Push.

7. To save the changes as a patch in a text file, hover the mouse pointer over the Submit/ Commit button and select Create Patch on the context menu.

   Tip Alternatively, use the Submit/ Commit drop-down list to select the Create Patch item.

   In the Create Patch dialog box, that opens, configure the patch creation .

8. If any error occurs when trying to commit, PhpStorm displays an error message. For example, you might have changed a file that has been already edited by another team member, or you might run into a branch conflict. In these cases, you need to merge edits , or update your local copy . The error messages are VCS-specific.

Tip Users of JetBrains TeamCity can obtain the TeamCity plugin for PhpStorm. Among the features of this plugin is Remote Run, which enables you to create a special *personal* build that does not affect the *real* build. Your changes are built and run through your test suite. If all tests are passed, your changes are automatically committed to version control.

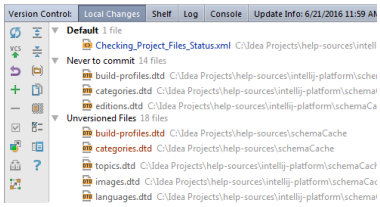
Checking Project Files Status

Apart from indicating the status of the currently opened file relative to the repository, PhpStorm provides you with an accumulated view of the project files' statuses.

To view the differences between the current state of the project files and the repository, do the following:

1. From the main menu, choose VCS | Refresh File Status.
2. Switch to the Version Control tool widnow and open the Local Changes tab.

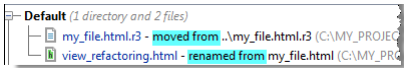
The status of each file is indicated by the color :



Copying, Renaming and Moving Files

There is no automatic renaming or moving files in the version control systems. Working with the clients, you have to manually copy a file to a new location, change its name, add to VCS and remove the old file. PhpStorm's refactoring features make it possible to easily move and rename files under version control, performing all actions involved in these functions.

In the Version Control tool window, next to the resulting files PhpStorm explicitly states: `renamed from <file name>` or: `copied from <folder name>`.



To copy, move or rename a file under version control

1. Select the desired file in the Project tool window, and perform refactoring procedures, as described in the sections:
   - Copy
   - Move
   - Rename

   All added and deleted files are placed to a changelist.

2. Commit changes to the repository.

Tip You can revert refactorings, using the Undo command.

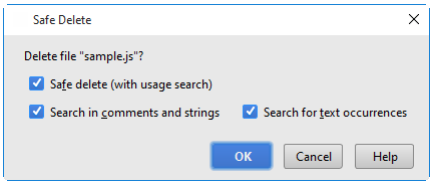
Deleting Files from the Repository

If you delete a file under version control, it still exists in the repository until deletion is committed. A deleted file is placed to the active changelist, and is displayed in grey font.

To delete a file

1. Select a file in any navigation view, and press `Delete`, or choose Delete on the context menu.

2. In the dialog that opens, you can opt to delete file without search for usages, or perform safe delete, to make sure that you are deleting an unused file, by checking Safe delete option. In this case, specify the refactoring options.

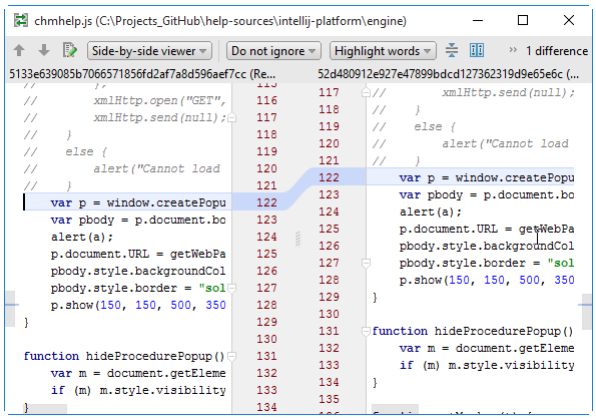


   The encountered usages are reported in the Usages Detected dialog box. You can view and correct these usages in the Safe Delete tab of the Find tool window. The deleted files are added to a changelist.

3. Commit changes to the repository.

Handling Differences

PhpStorm enables you to examine the differences between two revisions of a file, or between the current local copy and any repository version. Differences display in the Differences viewer . This window enables you to compare files and versions, navigate and search through the changes, copy and edit source code. Use the legend of the viewer to recognize modified, deleted and added lines of code.



This section describes how to:

- Compare file versions .
- Integrate contents of different versions into a file .
- Resolve conflicts between versions .
- Integrate your local copy of a project into a revision in the repository .

Comparing File Versions

VCS-specific integrations enable you to compare your local copy of a file with the versions stored in the repository.PhpStorm suggests several options:

- Compare your local copy with the repository version, to which you have last synchronized
- If somebody else has committed changes since your last update, compare your local copy with newest version
- Compare your local version with any repository version of a file .

For some VCS, it is possible to compare with a branch version. The differences display in the Differences viewer .

To compare with a repository version, to which you have last synchronized

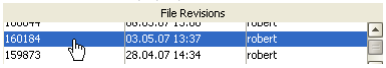
1. Select a file in the Project tool window, or open it in the editor.
2. Do one of the following:
   - On the main VCS menu, or on the context menu of a file, choose <VCS> | Compare with the same repository version.
   - Select a file in the Local Changes tab of the Version Control tool window, and choose Show Diff on the context menu.

To compare with the latest repository version

1. Select a file, or open it in the editor.
2. On the main VCS menu, or on the context menu of a file, choose <VCS> | Compare with the latest repository version.

To compare with the specified version of a file

1. Select a file, or open it in the editor.
2. On the main VCS menu, or on the context menu of a file, choose <VCS> | Compare with.
3. In the File Revision pop-up window, click the version to compare:

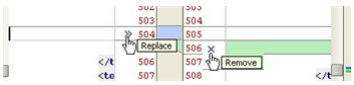


Tip Alternatively, use the History view of a file. Select the desired version, and choose Compare with Local on its context menu, or click on the toolbar.

Tip You can explore changes to binary files same way, as you do it for textual files. For example, use this feature to watch the changes made to images.

Integrating Differences

You might need to integrate contents of the different versions of a file, for example integrate changes of a certain branch into your local copy.



To integrate changes into your working copy

1. Select the desired file and compare it with a version .

2. In the Differences Viewer, use the *chevron* symbol for any block of lines that existed in the read-only copy and were changed or deleted in the local copy; and *x* symbol for the lines that were added in the local copy. Click these symbols to apply changes to the current local version of the file.

Integrating Project

You might need to integrate your local version of a project into a certain revision of that project in the repository. For this purpose PhpStorm provides Integrate Project command.

Integrating project is only available for the projects that contain directories associated with Perforce or Subversion. If neither of these two VCS is involved in project, the Integrate Project command is disabled in the menu.

To integrate different sources into one project

1. On the main menu, choose VCS | Integrate Project. The Integrate dialog appears.
2. Select Perforce or Subversion tab, and configure merge options as required. Refer to the respective procedural topics for details.

Resolving Conflicts

Depending on the type of your version control system, conflicts arise in different situations.

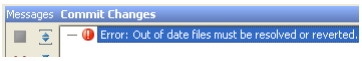
Non-Distributed Version Control Systems

Working in a team, you might come across a situation, when somebody commits changes to a file you are currently working on. Conflicts can also arise on merging branches, applying patches or unshelving changes. If the changes are not overlapping, the files are merged automatically, but when there are changes to the same lines, you will need to resolve such conflict yourself.

When you try to edit a file that has a newer version on the server, PhpStorm informs you about that, showing a message pop-up window in the editor:

In this case you should update your local version prior to changing the file, or merge changes.

If you try to commit a file that has a newer repository version, an error is reported in the Messages tool window:



The commit behavior is regulated by the Create changelist on failed commit drop-down list in the Confirmation page of the Settings / Preferences dialog.

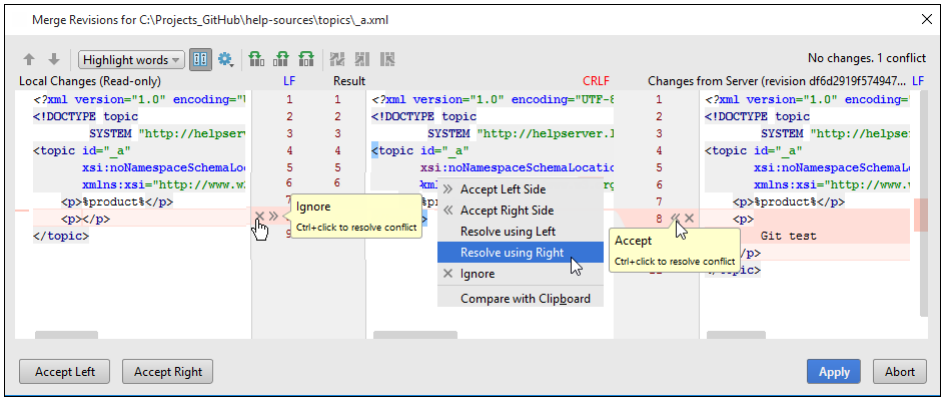
If you synchronize a file, that already has local changes, with a newer repository version, committed by somebody else, a conflict occurs. The conflicting file gets status Merged with conflicts. The file remains in the same changelist in the Local Changes tab of the Version Control tool window, but its name is highlighted red. If the file is currently opened in the editor, the file name on the tab header is also highlighted red.

Distributed Version Control Systems

Under Git, conflicts arise during merge , rebase , cherry-pick , unstash , or apply patch .

Conflict Resolution Tool

PhpStorm suggests a tool for resolving conflicts locally. This tool consists of three panes. The left pane shows the read-only local copy; the right pane shows the read-only version checked in to the repository. The central pane shows a fully-functional editor with the results of merging and conflict resolving. Initially, the contents of this pane is the same as the base revision of the file, that is, the revision from which both conflicting versions derived.



To resolve conflicts, follow these general steps

1. Select the conflicting file in the Version Control, Update Version Control, or Project tool window, or open it in the editor.
2. On the context menu, under the node of your VCS integration, choose the VCS-specific merge command.
3. Perform edit in the merge tool to apply or discard changes, as described in the procedures below.

To apply or discard a change

1. In the left or right pane, select the change to be merged. Use arrow and/or find buttons on the toolbar to navigate to the desired location.
2. Accept or reject changes, clicking the chevron or X button respectively.
3. Click Apply.

To apply all non-conflicting changes

– On the toolbar of the merge tool, click . It is useful to automatically accept any changes that are not in conflict, and spend your time working on conflicting changes.

Running PhpStorm as a Diff or Merge Command Line Tool

In this topic:

- Overview
- Enabling invocation of PhpStorm operations from the command line
- Comparing files using PhpStorm as a command line tool
- Merging files using PhpStorm as a command line tool
  - Passing three arguments to merge tool

Overview

Besides using PhpStorm as an Integrated Development Environment, you can use it as a command line tool for comparing and merging files.

PhpStorm executable is platform-dependent:

- Windows: `PhpStorm XX\bin\phpstorm.exe` or `PhpStorm XX\bin\phpstorm.bat`
- UNIX: `PhpStorm XX/bin/phpstorm.sh`
- OS X: `/Applications/PhpStorm XX.app/Contents/MacOS/phpstorm`
  To add the launcher to your path, add its containing directory `/Applications/PhpStorm.app/Contents/MacOS`.

However, for OS X and UNIX, one should create a wrapper script, since this helps avoid some drawbacks related to the usage of PhpStorm launcher.

Enabling invocation of PhpStorm operations from the command line

For OS X and UNIX platforms, we recommend creating the wrapper script, or the command line launcher to integrate PhpStorm with your shell. Then, you need to ensure that the created launcher script is within the search path of your shell.

On Windows, we recommend to add the path to PhpStorm to the environment variable `Path`. Everything is done outside of PhpStorm, with the PhpStorm executable.

Note that if you have specified location of the PhpStorm executable as a `Path` environment variable, the command will work no matter which directory you are currently in.

To enable invoking PhpStorm operations from the command line, follow these steps

- On OS X or UNIX:
  1. Make sure PhpStorm is running.
  2. On the main menu, choose Tools | Create Command-line Launcher. The dialog box Create Launcher Script opens, with the suggested path and name of the launcher script. You can accept default, or specify your own path.
     Make notice of it, as you'll need it later.
  3. Outside of PhpStorm, add the path and name of the launcher script to your path.

- On Windows:
  - Specify the location of the PhpStorm executable in the `Path` system environment variable. In this case, you will be able to invoke the PhpStorm executable and other PhpStorm commands from any directory.

Comparing files using PhpStorm as a command line tool

To compare two files using PhpStorm as a diff command line tool

1. Enable invoking PhpStorm operations from the command line .
2. Type the following command at the command line prompt:

```
<PhpStorm launcher(Windows) or wrapper script (MacOS or UNIX)> diff <path to file1> <path to file2>
```

where `file1` is your local copy, `file2` is the repository version.

For example:

```
phpstorm diff README.md.bak README.md
```

Merging files using PhpStorm as a command line tool

Most often you need to merge three versions of the same file: your local version, the version in the repository or in the upstream, and the base revision, which is the origin for the two diverged versions.

To merge files using PhpStorm as a command line tool

1. Enable invoking PhpStorm operations from the command line .
2. Type the following command at the command line prompt:

```
<PhpStorm launcher(Windows) or wrapper script (MacOS or UNIX)>
merge <path to file1> <path to file2> <path to file3> <path to output>
```

where `file1` is your local copy, `file2` is the repository version, `file3` is the base revision for `file1` and `file2`, and `output` is the file to save the merge results in (optional).

Passing three arguments to merge tool

It is possible to pass just three arguments to the merge tool:

`<path to file1> <path to file2> <path to output>`.

In this case, the contents of the output will be taken as the base revision:

```
<PhpStorm launcher> merge <path to file1> <path to file2> <path to output> <path to output>
```

See the example in this blog to learn how to use PhpStorm diff and merge tool with Git.

Handling Issues

With PhpStorm, you can connect your check-in comments with the bug tracker or any issues data base and navigate from committed changes to the issues related to these changes.

To enable this navigation, you need to specify a so called issue navigation pattern, which means:

1. Figure out an issue ID pattern, that is, a format according to which you will reference issues in commit messages, and define this issue pattern using a regular expression.
2. Define the link to the referenced issue by combining the URL address of your tracking system and a regular expression to identify the issue ID.

In other words, an issue navigation pattern maps an issue ID pattern in commit messages and URL addresses of referenced issues. As soon as PhpStorm encounters a match to the issue ID pattern in a commit message, the match is displayed as a link in the Changes and Version Control tool windows. If you mention several issues, all of them will show up as links. Clicking such link opens the matching issue in the browser according to the defined link.

On this page:

- Example
- Enabling navigation from commit messages to issues related to them
- Navigating from a commit message to the related issues

Example

Issue ID pattern	The regular expression that defines the format in which issues are referenced in commit messages. `[A-Z]+\-\d+` This regular expressions matches all character strings that consist of two substrings separated by an n-dash character: 1. Substring 1: An unlimited number of upper case alphabetic characters. 2. Substring 2: An unlimited number of digital characters.
Issue link pattern	A combination of the URL address of your issue tracking system and a regular expression that identifies issues in it. `http://mytracker/issue/$0` Here `$0` indicates a back reference to the entire match. This means that as soon as PhpStorm detects a match in a commit message, it is added to the URL address of the tracker as is.
Matching issue ID	PhpStorm detects the following reference to an issue in the commit message of interest: `MYPROJECT-110`
Composed issue link	In accordance with the above issue navigation pattern, the detected matching reference is added to the URL of the tracker as is, so the link to the referenced issue is composed as follows: `http://mytracker/issue/MYPROJECT-110`

To enable navigating from commit messages to issues related to them

1. Open the Settings dialog box , and click Issue Navigation under the Version Control node.
2. In the Issue Navigation dialog box that opens, configure a list of issue navigation patterns by setting correspondence between issue patterns in commit messages and URL addresses of referenced issues.
   - If you are using JIRA or our bug tracking system YouTrack , click the Add JIRA pattern or Add YouTrack Pattern respectively, and type the URL to the installation of bug tracking system in question.

     PhpStorm adds the regular expression that defines such pattern automatically.
   - For other issue tracking systems, click the Add button to create a new entry or select an existing entry and click the Edit button. In the Add Issue Navigation Link dialog box that opens, specify the following:
     1. The regular expression that defines the issue pattern in a commit message.
     2. The replacement expression that defines the URL to access the corresponding referenced issue.
   - To remove an issue navigation pattern, select it in the list and click Remove .

To navigate from a commit message to the related issues

1. Open one of the following views:
   - Local Changes, Incoming, or Log tab of the Version Control tool window.
   - History tab of the Version Control tool window.

- Changes Browser .

2. Find the commit of interest and click the hyperlink to the related issue.

Managing Changelists

This section describes how to:

- Create , delete and rename changelists.
- Assign active changelists .
- Group items in a changelist .
- Move files between changelists .
- Jump from an item in a changelist to the corresponding source code in the editor .

Assigning an Active Changelist

Active changelist is the one to which the changed files are added automatically. The name of the active changelist is highlighted in bold font.

To assign an active changelist

1. Select a changelist in the Version Control tool window.
2. On the context menu of the selected changelist, click Set Active Changelist.

To create a new changelist

1. In the toolbar of the Version Control tool window, click + button.

   Tip You can also use one of these alternatives:
   - Right-click anywhere in the Local Changes tab of the Version Control tool window and choose New Changelist on the context menu.
   - Press `Alt+Insert`.

2. In the New Changelist dialog, specify the name of the new changelist, and optional comment.

3. Click OK.

Deleting a Changelist

When a changelist is deleted, all changes are moved to the active changelist.

To delete a changelist

1. In the Version Control tool window , select a changelist to be deleted.

2. In the toolbar of the Version Control tool window, click —. Alternatively, right-click the changelist node and choose Delete Changelist on the context menu, or just press `Delete` key.

3. If the changelist is not empty, you are prompted to confirm deletion and move uncommitted items to the active changelist. If you attempt to delete an active changelist, you are prompted to specify another
   If Perforce is used for a certain directory, deleting the default changelist is not allowed.

Grouping Changelist Items by Folder

Within each node of the Version Control tool window , you can display the modified files as a flat list, or as a directory tree.

To toggle grouping items by directories, do one of the following

- On the toolbar of the Version Control tool window, click .
- Use `Ctrl+P` keyboard shortcut.

To move items between changelists in the Version Control tool window

1. In the Version Control tool window, select one or more desired items in a changelist. Use the `Ctrl` and `Shift` keys for multiple selection.

2. Choose Move to Another Changelist on the context menu of the selection.
   You can also use one of these alternatives:
   - Click the Move to Another Changelist button on the toolbar of the tool window.
   - Press `F6`.
   - Drag the selected items to the target changelist.

3. In the Choose Changelist dialog box, specify the changelist to move the selected items to:
   - If the target changelist exists, click the Existing Changelist option and select the desired changelist from the drop-down list.
   - To create a changelist, click the New Changelist option, type the name of the new changelist, and optionally provide a description.

To navigate from an entry in the changelist to the source code

1. In the Version Control Tool Window, expand a changelist and select the desired entry.
2. On the context menu of the selection, choose Jump to Source, or press F4.

To rename a changelist

1. Select a changelist in the Version Control Tool Window.
2. On the context menu, choose Edit Changelist.
3. In the Edit Changelist dialog, specify the new name and optional description, and click OK, or use the `Shift+F6` keyboard shortcut..

Refreshing Status

This feature is helpful if a server, that holds the sources of the project files, is down. The statuses of files and directories become irrelevant, and you are unable to work with the local copy of the project. To avoid such situation, you need to refresh the status of your source files.

When this command is applied, PhpStorm refreshes the status of each file, no matter whether the file has been changed from PhpStorm itself or using any other application.

When working under Perforce control, you can run refresh in two modes:

- Standard Refresh takes into consideration only the changes made through the PhpStorm integration with Perforce. This improves performance because does not require connecting to the server. However, this approach does not let you know about the changes made outside PhpStorm, for example, right through the `p4v client` application.
- Force Refresh considers all the changes made to project, both from PhpStorm and from any other application, for example, right through `p4v client`.

The status of a file is refreshed only in accordance with the changes in your local workspace. Any changes on the server can be reflected only through synchronization .

To refresh the status of files in your project, do one of the following

- On the main menu, choose VCS | Refresh File Status.
- In the Version Control tool window, press `Ctrl+F5`.
- In the Version Control tool window, click the Refresh toolbar button.
- For Perforce integration, do one of the following:
  - To run Standard Refresh, click the Refresh toolbar button or press `Ctrl+F5`.
  - To run Force Refresh, click the Force Refresh toolbar button.

Reverting Local Changes

When you modify, add, or delete files, which are under version control, you are always able to revert such changes, rolling back the file's contents to what it was before the last successful update, check out, or commit.

The exact name of the command (revert or roll back), and the type of the action performed when you revert changes, depend on the file status and VCS used for a particular directory.

To revert local changes, do one of the following

- In the Local Changes tab of the Version Control tool window, select one or more items in the relevant changelist , then choose Revert on the context menu of the selection or click the Revert button on the toolbar of the Version Control tool window.
- Select the file to be reverted and choose the relevant VCS-specific revert (roll back) command on the VCS | <VCS> menu.

Reverting to a Previous Version

You can restore any previous version of a file, using the History view of a file. So doing, the current content of the file is replaced with the copy of the older content. After such operation, you have to commit the file to bring the repository up to date.

The exact name of the command (revert or roll back) depends on the specific VCS.

To revert a file to its previous version

1. Select the desired file in the Project tool window, and open its history .
2. In the History tab, select the desired revision.
3. On the context menu of the selected entry, choose Get, or click the button on the toolbar.

Shelving and Unshelving Changes

Shelving is temporarily storing not yet committed changes in a dedicated shelf .

Unshelving is returning postponed changes from a shelf to a pending changelist.

With PhpStorm, you can shelve and unshelve both separate files and entire changelists or shelves.

Unshelved changes can be filtered out from the view or removed from the shelf. Once shelved, a change can be applied as many times as you need by unshelving and subsequently restoring it on the shelf.

In the Git integration , in addition to shelving and unshelving "stashing" and "unstashing" are supported respectively. These features have much in common, the only difference is in the way patches are generated and applied.

- Patches with stashed changes are generated by Git itself. To apply them later, you do not need PhpStorm.
- Patches with shelved changes are generated by PhpStorm. Normally, they are also applied through the IDE. Applying shelved changes outside PhpStorm is also possible but requires additional steps.

In this section:

- Shelving Changes
- Unshelving Changes
- Restoring Unshelved Changes
- Filtering Out and Removing Unshelved Changes

Shelving Changes

On this page:

– Basics

– Shelving changes

– Shelving base revision automatically

Basics

Shelving is temporarily storing not yet committed changes in a dedicated shelf . This may be helpful, for example, when you need to switch to another urgent task, but you are short of time to bring your current code to a certain condition.

With PhpStorm, you can shelve both entire changelists or separate files.

Once shelved, a change can be applied as many times as you need by unshelving and subsequently restoring it on the shelf.

When using Git or Mercurial integration, it may be useful to have PhpStorm always shelve the base revisions of files that are under Git or Mercurial version control.

By default, PhpStorm always "remembers" the last commit hash. However, this information is not sufficient if the history has been changed since the last commit as a result of running the Rebase operation. In this case, having a copy of the base revision may help.

Shelving changes

1. In the Local Changes tab of the Version Control Tool Window tool window, select the files or changelist to put to a shelf . On the main Version Control menu or on the context menu of the selection, choose Shelve changes.
2. In the Shelve Changes dialog box, review the list of changed files and make sure that the files to be put to a shelf are checked out.

   Tip You can specify the desired changelist immediately in the dialog box: from the Changelist combo box, select the desired changelist.

3. In the Commit Message text box, type the comment to identify the shelf where the changes will be stored.
4. Click Shelve changes.

Shelving base revision automatically

1. Open the Settings dialog box, and click Version Control.
2. On the Version Control page, select the Store on shelf base revision texts of files under DVCS check box.

Unshelving Changes

On this page:

- Basics
- Unshelving changelists and files
- Applying external patches

Basics

Unshelving is moving postponed changes from a shelf to a pending changelist.

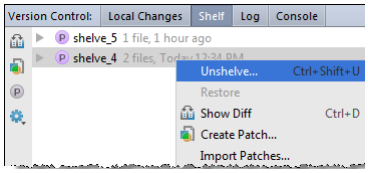
With PhpStorm, you can unshelve both entire shelved changelists and separate files.

Unshelved changes can be filtered out from the view or removed from the shelf. Once shelved, a change can be applied as many times as you need by unshelving and subsequently restoring it on the shelf.

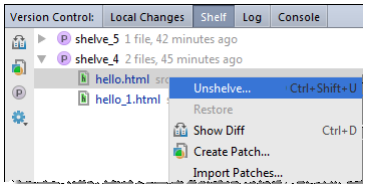
You can also import patches created inside or outside PhpStorm and apply them as shelved changes.

Unshelving changelists and files

1. In the Shelf tab of the Version Control tool window , select the changes to unshelve:
   - To unshelve an entire changelist, just select its name.



   - To unshelve the changes from a specific file, expand the relevant changelist node and select the desired file.



2. On the context menu of the selection, choose Unshelve.
3. In the Unshelve Changes Dialog that opens, specify the changelist you want to restore the unshelved changes to.
   - To add the changes to an existing changelist, choose Existing Changelist and select the relevant changelist from the drop-dow list.
   - To create a new changelist and restore the changes from the shelf in it, choose New Changelist.
     - In the Name text box, type the name of the changelist to be created.
     - In the Comment text box, type an optional description of the new changelist.
     - To have PhpStorm automatically give the active status to the new change list immediately after the changes are restored in it, select the Make this changelist active check box. When this check box is cleared, the current active changelist remains active. See Changelist for details.
     - To have PhpStorm preserve the context of the task associated with the new changelist on its deactivation and restore the context when the changelist becomes active, select the Track context: check box. See Managing Tasks and Context for details.
4. To have PhpStorm still display already unshelved changes in the Shelf tab so that you can apply them once more if necessary, clear the Remove successfully applied files from the shelf check box.
   When this check box is selected, the changes are not displayed in the Shelf tab after they are unshelved.
5. Click OK to start the process.
   If conflicts occur between the patched version and the current version, resolve them as described in Resolving Conflicts . Select the changes in both versions to be merged to the resulting file and click Apply.

Applying external patches

You can also import patches created inside or outside PhpStorm and apply them as shelved changes.

1. In the Shelf tab of the Version Control tool window, choose Import Patches from the context menu.
2. In the dialog box that opens, select the patch file to apply. The selected patch appears in the Shelf tab as a shelf.
3. Select the newly added shelf with the patch and choose Unshelve Changes from the context menu of the selection.

Restoring Unshelved Changes

With PhpStorm, you can apply changes as many times as you need by restoring them to the shelf upon unshelving. You can restore both separate files and entire shelves. Unshelved changes are available for re-use until they are explicitly removed, see Filtering Out and Removing Unshelved Changes .

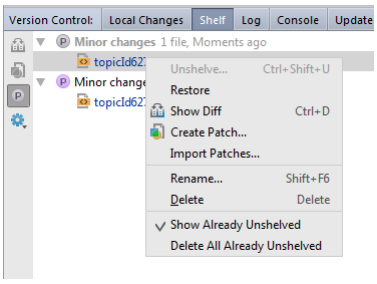
By default, upon unshelving, a change is no longer shown on the list of available shelved changes, therefore you first need to have PhpStorm display it in the Shelf tab.

To restore unshelved changes

1. In the Shelf tab of the Version Control tool window, click the toolbar button or choose the Show Already Unshelved option on the context menu. All previously applied shelves are again on the list and are available for unshelving.

2. Select the desired files or shelf to restore.

3. On the context menu of the selection, choose Restore.



Filtering Out and Removing Unshelved Changes

Most shelved changes are applied only once. By default, upon unshelving a change is no longer shown on the list of available shelved changes. This helps you keep your list of available shelved changes more easier to use. However you can return an applied shelf to the list it at any time, restore and unshelve it again as many times as necessary.

Shelves that are not likely to be applied in the future anymore, can be permanently removed from the list.

On this page:

- Removing a single change
- Returning unshelved changes to the list
- Hiding unshelved changes
- Removing unshelved changes permanently

Removing a single change

You can permanently delete any change no matter whether it has been already unshelved or not.

- To remove a shelved change permanently, select it in the list and choose Delete on the context menu of the selection.

Returning unshelved changes to the list

- In the Shelf tab of the Version Control tool window, choose Show Already Unshelved on the context menu.

Hiding unshelved changes

- In the Shelf tab of the Version Control tool window, choose Hide Already Unshelved on the context menu.

Removing unshelved changes permanently

1. In the Shelf tab of the Version Control tool window, choose Show Already Unshelved on the context menu.
2. Select the desired file or changelist.
3. On the context menu of the selection, choose Delete All Already Unshelved.

Updating Local Information

The Update command enables you to synchronize the contents of your local files with the repository. You can invoke this command on:

- Single or multiple files and directories
- An entire project

Depending on the updating options, the update procedure may take place silently. If all files are up-to-date, you will be notified about that. Otherwise, the Update Info tab opens in the Version Control tool window where you can group the information as required.

On this page:

- Updating files and folders
- Updating a project
- Grouping update information by packages or changelists

Updating files and folders

1. Select one or more files and folders to be updated in any navigation view (for example, in the Project tool window .
2. On the main Version Control menu, or on the context menu of the selection, choose <VCS>| Update.
3. In the Update dialog specify the update options, which are different for the supported version control systems, and click OK.

Updating a project

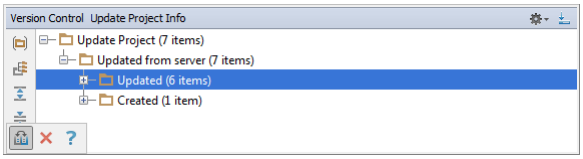
1. Do one of the following:
   - On the main menu, choose VCS | Update project
   - Press `Ctrl+T`.
   - On the main menu, click .
2. In the Update dialog, click the tab for your version control system.
3. Specify the update options, which are different for the supported version control systems and click OK.

Grouping update information by packages or changelists

To group the update information by namespaces or changelists, use the Group By Package and Group By Changelist buttons on the toolbar of the Update Info tab.

Note the difference in the appearance of the tab:

- Grouped by namespaces



- Grouped by changelists

Grouping by changelists is not available if the project is under Git or Mercurial control.

Using Patches

This section describes how to:

- Create patches .
- Apply patches .

Applying Patches

Postponed changes stored in a patch file can be applied to the target file or directory later. If the source code was edited after creating the patch, conflicts may arise. PhpStorm suggests a handy way to resolve such conflicts and merge the patch with the changes.

When a patch is opened, PhpStorm detects files with the same name as the modified files. For each detected file, PhpStorm compares the path to it relative to the base directory, with the path from the patch, and chooses the closest match. If no matching path is found, the file is considered to be located in the project base directory and is highlighted in red.

You can apply changes to files stored in different locations from those specified in the patch by mapping an arbitrary directory as the base one, or stripping off the leading directories.

To apply a patch

1. On the main menu, choose VCS | Apply patch.

2. In the Apply Patch dialog box that opens, specify the fully qualified name of the patch file. Type the name manually or click the Browse button and locate the desired patch file in the Select Patch File dialog box.

   You can also drag and drop a file or an email attachment to the Apply Patch dialog, and it will be selected automatically.

3. Configure the patch presentation layout. To have changes shown in a flat view, press the Group by Directory toolbar button .

   Release the button to have changes shown in a directory tree view.

4. To have a change applied, select the check box next to it.

5. To have a change applied to a modified file that has been moved to another location, specify the new file location.
   - To map another base directory, select the desired file, directory, or a group files/directories and click the Map base directory toolbar button . In the dialog that opens choose the directory relative to which file names will be interpreted.
   - To remove leading directories from the path, click the Strip Directory toolbar button as many times as many leading directories you need to strip. The number of removed slashes is indicated in square brackets.
   - To revert the last strip directory action, click the Restore Directory toolbar button . Click the button as many times as many previously stripped leading directories you need to restore.
   - To revert all the strip directory actions in the selection, click the Reset Directories toolbar button .
   - To have all the leading directories stripped and have the changes applied to the file with the specified name in the base directory, click the Remove Directories toolbar button .

6. To view the differences and possible conflicts between your local working copy, the repository version, and the patch in the Differences viewer for files , select the desired change and click the Show Differences button .

7. To resolve conflicts between the patched and the current versions, if any, in both versions select the changes to be merged to the resulting file, and then click Apply.
   Note that you apply a patch that contains a lot of files and causes numerous conflicts, you can cancel applying the patch by clicking Abort, and then select whether you want to abort, skip or cancel the remaining conflicts.

Creating Patches

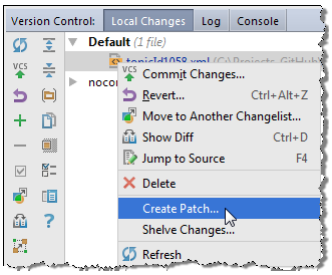
PhpStorm suggests two ways of creating patches:

- On the base of revisions, either local or committed to the repository
- On the base of revisions stored in the local history.

To create a patch file

1. In the Local Changes tab or the Repository tab of the Version Control tool window, select a change or changelist you want to create a patch for.

2. On the main Version Control menu or on the context menu of the selection, choose Create patch.



3. In the Create Patch dialog box that opens, review the list of changed files, and make sure that the files to be included in the patch are selected.

   Tip You can specify the desired changelist immediately in the Create Patch dialog box: click Changelist combo box in the upper right corner, and select the desired changelist.

4. Add a commit comment.

   Tip As you type, PhpStorm checks the spelling and highlights erroneous words.

   This functionality is available if the 'Spelling' code inspection is enabled.

5. Click Create patch.

You can also create patch on the base of your local history. To do that, open the local history view for the desired directory, file or code fragment, as described in the section Using Local History , right-click the desired revision, and choose the Create Patch command on the context menu, or click the create patch button on the toolbar.

Viewing Changes Information

PhpStorm provides various ways to keep track of the changes that you and your teammates introduce to the source code. You can:

- View local changes: use the editor where you create your source files to view, explore and navigate through your local changes within a single file.

  Use the Local Changes tab of the Version Control tool window to examine local changes throughout the entire project.
- View changes made by other team members:

  Use the Committed and Incoming tabs to view the changes committed to the repository by the other team members .
- Viewing history:

  Browse through the changes , filtering the range you are interested in.

  Explore history of a file in a dedicated view.

  Show annotated view of a file to drill deeper into the file history and find changes information for each line of code.

Browsing Changes

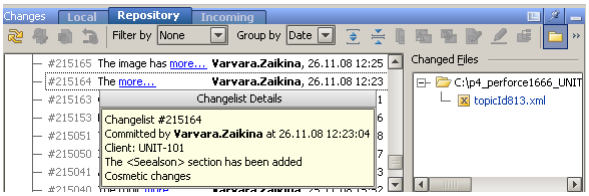
Browsing changes feature helps you see all changes during the history cache period, or confined to a certain period, user, or changelist .

All changes for the history cache period are displayed in the Repository tab of the Version Control tool window, Changelists pane.

To browse all changes in the history cache

1. In the Version Control tool window, switch to the Repository tab.

2. Click the refresh button on the toolbar, or choose Refresh on the context menu.



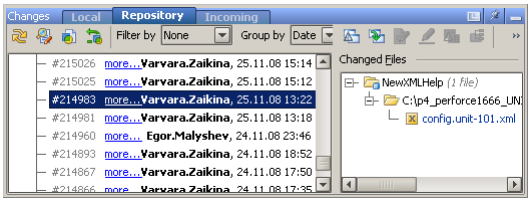
To browse changes to files or folders

1. Select the desired file or folder in the Project window.

2. On the Version Control menu or on the context menu of the selection, choose <VCS> | Browse Changes . The Change Search Criteria dialog box is opened.

3. Specify the search criteria (type the necessary filter value or press the ellipsis button to set the value for Author, Date, and Change filters) and click OK.
   If you do not specify any filtering criteria, you will be prompted either to show recent changes only, or view all history of the entire project.

   Setting too wide filter values for a large project might require considerable time for the changes to be found.



Using Change Markers to View and Navigate Through Changes in the Editor

As you modify a file that is under version control, all changes are highlighted in the editor with the change markers that appear in the left gutter next to the modified lines, and show changes introduced since the most recent synchronization with the repository. On committing a file to the repository, the change markers disappear.

The changes you introduce to the text are color-coded:

- line added.
- line changed.
- line deleted.

A special toolbar is associated with each change marker, enabling you to perform the following operations:

- Navigate between changes using the and buttons in the change markers or the `Ctrl+Shift+Alt+Down` or `Ctrl+Shift+Alt+Up` keyboard shortcuts. See Navigating between Changes for details.
- Compare the current version with the last committed one using .
- Revert changes using .
- Put the previous version of the changed lines to the clipboard using .

To show the change marker toolbar

1. Point to the desired change marker. The mouse pointer changes its shape:

2. Click the change marker. The toolbar and the previous contents of the line are displayed:

Viewing Details of Changes

Besides navigating through your local changes within a file in the editor, you can view and examine these changes compared to the base revision of the file.

This functionality is provided through a dedicated Change Details pane in the Local Changes tab of the Version Control tool window.

The pane consists of two areas:

- Area (1) shows the affected code as it was in the base revision.
- Area (2) shows the affected code as it is after the change is introduced.

In each area, PhpStorm numbers both changes themselves (4) and the lines (3) affected by them. The changes introduced to the code are color-coded:

- line changed (5).
- line added (6).
- line deleted (7).

A special toolbar (8) enables you to perform the following operations:

- Navigate between changes using the Next Change and Previous Change buttons.
- Expand or narrow the context of a change using the More/Less Lines button .
- Configure the appearance of the pane using the Settings button .

In this section:

- Opening the dedicated Changes Details pane
- Examining details of changes in a file
- Configuring the appearance of the pane
- Closing the Change Details pane

To open the dedicated Change Details pane

1. Open the Version Control tool window (`Alt+9`), and then switch to the Local Changes tab.
2. Click the Change Details toggle button on the toolbar. The Change Details pane opens.

Note that rhe Change Details toggle button remains pressed until the pane is closed.

To examine details of changes in a file

1. Open the Version Control tool window (`Alt+9`), and then switch to the Local tab.
2. Select the file in question, and then open the Change Details pane, if it is not opened. The base revision and the local version of the selected file are opened with the differences numbered and highlighted.
3. Examine the details of each change:
   - To move to the next updated piece of code, click the Next Change button .
   - To return to the previous updated code fragment, click the Previous Change button .
   - To expand or narrow the context of an updated code fragment, position the cursor at the change in question, click the More/Less Lines button , and then specify the number of lines to be shown above and below the current code fragment.

To configure the appearance of the Change Details pane

1. Click the Settings button on the toolbar of the pane.
2. On the context menu, that opens, specify how you want the pane split:
   - To have the pane split horizontally, so the base revision is shown in the upper part and the locally updated version is shown in the bottom part of the pane, select the Top/Bottom option.
   - To have the pane split vertically, so so the base revision is shown in the left-hand part and the locally updated version is shown in the right-hand part of the pane, select the Left/Right option.
3. To have the soft wraps (or word wraps) used, select the Use soft wraps option.

To close the Change Details pane

- Click the pressed Change Details toggle button on the toolbar.

Working with Annotations

In this topic:

- Basics
- Showing and hiding annotations
- Configuring the amount of information shown in the annotations pane
- Annotating previous revisions
- Viewing differences between revisions
- Navigating to log

Basics

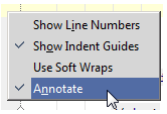
Annotation is a form of file presentation that shows detailed information for each line of code. In particular, for each line you can see the version from which this line originated, the user ID of the person who committed this line, and the commit date. The annotated view helps you find out who did what and when, and trace back the changes.

Annotating lines of code is available for ClearCase, TFS, Mercurial, Git, CVS, Perforce and Subversion.

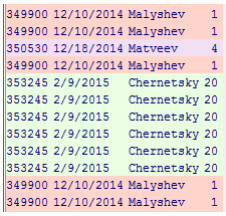
The Annotate command appears in the VCS-specific nodes of the Version Control menu, file context menus, and the File History view. This command toggles between the plain and the annotated view of a file.

Showing and hiding annotations

1. Open the desired file in the editor.
2. To show annotations, right-click the left gutter, and select Annotate:



When annotations are enabled, the left gutter looks similar to the following example:



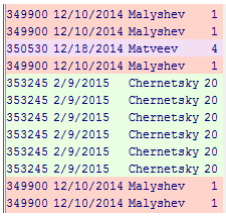
The annotations pane provides the following information:

- The number of the changelist within which the annotated change was checked in.
- The date when the annotated change was checked in.
- The name of the user who checked in the annotated change.
- The revision number of the current file.

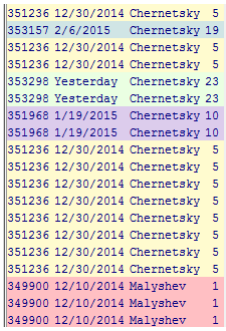
3. To hide annotations, right-click the annotations gutter, and choose Close Annotations.

Configuring the amount of information shown in the annotations pane

1. Enable annotations and right-click the annotations gutter.
2. Select View in the context menu and enable or disable the following options according to the type of information you want to be displayed:
   - Revision: select this option if you want to see the number of the changelist within which the annotated changes were checked in.
   - Date: select this option if you want to see the date when the annotated changes were checked in.
   - Author: select this option if you want to see the name of the user who checked in the annotated changes.
   - Commit number: select this option if you want to see the revision number of the current file.
   - Colors: use this control to toggle between the following highlighting modes:
     - Author: select this option if you want to highlight changes made by different authors with different colors:

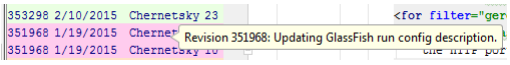


     - Order: select this option if you want annotation colors to indicate how recently a change was made. The entire file history is divided into several time periods containing an equal number of commits, and each time period is assigned its own color. The most recent changes are highlighted in green, while the oldest changes are highlighted in red:



  - Hide: select this option if you do not want to use any color highlighting. In this case, all annotations will be displayed in gray.
- Names: use this control to select how user names will be presented. The following options are available:
  - Last name
  - First name
  - Full name

3. To view a commit message for an annotated change, hover the mouse cursor over an annotation. A tooltip will appear showing the commit message for the corresponding change:



Note that the amount of information displayed in the tooltip depends on the version control system you are using and is not affected by the annotation settings .

Annotating previous revisions

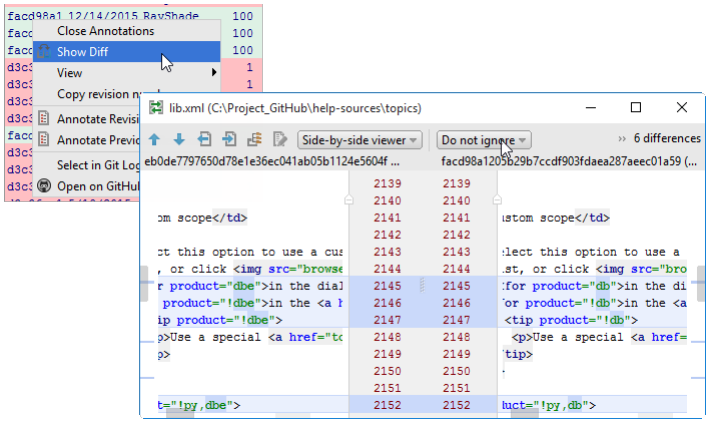
PhpStorm lets you annotate not only the current revision of the file, but also it's previous revisions. The following options are available:

- Annotate Revision: this option is useful if you want to check what a file looked like after a particular change was committed. To do this, right-click this change and select Annotate Revision from the context menu.
- Annotate Previous Revision: this option is useful if you find yourself in a situation when the last change in a particular line is meaningless, for example if all that was changed is code formatting. In this case, you can check what the previous revision of the file looked like. To do this, right-click a change and select Annotate Previous Revision from the context menu.

You can also annotate a particular file from the file history view . In the History tab, select the file version you want to review, right-click the corresponding line and select Annotate from the context menu.

Viewing differences between revisions

Position the cursor on an annotation, right-click it and select Show Diff from the context menu. PhpStorm opens the Differences viewer for files where you can view what changed between the annotated revision of the file and its previous revision:



Navigating to log

Position the cursor on an annotation, right-click it and select Select in Git log from the context menu. PhpStorm opens the Log Tab where you can view the specific revision.

The same result can be achieved by choosing the Copy revision number command from the context menu. Using this revision number, you can find the desired revision in the log.

Viewing Changes History for a File or Selection

VCS integrations of PhpStorm enable you to inspect the changes made to files, or even fragments of source code. Show History and Show History for Selection commands are available on the main VCS menu, and on the context menus of the files, under the respective VCS nodes.

Results of the Show History command are displayed in a dedicated tab of the Version Control Tool Window . For the different version control systems, the set of toolbar buttons in the History tab may slightly vary. Results of the Show History for Selection command are displayed in a dedicated window, in the form of a differences viewer.

Viewing the History of a File

1. Do one of the following:
   - Open the file in the editor. Then on the main VCS menu or on the context menu of the editor tab, choose <directory VCS> | Show History.
   - In the Project tool window, select the file in question and choose <directory VCS> | Show History on the context menu of the file.

   The History tab for this file appears in the Version Control tool window, the name of the file is shown on the title bar of the tab.
   File revisions are shown in regular font. The revision that currently exists in your working copy, is shown in bold font.
2. Click the columns to filter the list of changes you have made to the selected file.
3. Click the relevant buttons on the toolbar to compare with local version, get the selected version from VCS, annotate a file, refresh the revisions list, and show the list as a tree.
4. To view the other files updated in a revision, select the revision in question and choose Show All Affected Files on the context menu. The Paths Affected in Revision dialog box opens showing a list of all the files that were modified in the selected revision.

Viewing the History of a Selection

1. In the editor, select the fragment of source code to view the history for.
2. Choose <directory VCS> | Show History for Selection on the main VCS menu, or on the context menu of the selection.

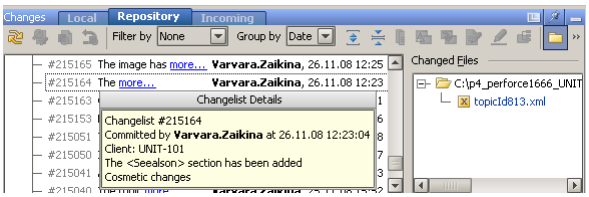
Viewing Changes Made by Other Team Members

Use the Repository and Incoming tabs of the Version Control tool window to view changes introduced by the team. Repository tab shows all changes that you and your colleagues have committed to the repository; the Incoming tab shows only those committed changes, which are not yet taken to your local working copy. PhpStorm informs you about new changes in the repository by the icon on the status bar.

To view all changes committed to the repository

1. In the Version Control tool window, select the Repository tab.

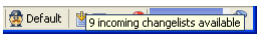
2. Click refresh button on the toolbar, or choose Refresh on the context menu.



To view incoming changes

1. In the Version Control tool window, select the Incoming tab.

2. Click refresh button on the toolbar, or choose Refresh on the context menu. Alternatively, click the incoming changelists icon in the status bar:



Viewing File Status

PhpStorm provides facilities for monitoring the status of project files relative to the repository. The status of a file reflects which operations have been performed on the file locally since the last synchronization with the repository. Using this functionality requires the following prerequisites:

- Integration with a version control system is enabled.
- The entire project or its specific directories are associated with this version control system.

You can monitor the status of a project in the following ways:

- In any interface component by the color used to display the name of a file.
- In the Version Control tool window, Local tab, which by default shows files grouped according to their status. The name of nodes depend on the VCS you are using.

  Tip To have files grouped by directories, click the icon.

- For Git and CVS integrations, in the Version Control tool window, Check Project Status Info tab, via the Check Project Status command.

Accessing the Authentication to Server Dialog

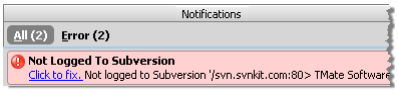
For a number of reasons of various nature you may be not authenticated to the remote server while the authentication dialog box does not appear, as you might expect. Instead, PhpStorm displays the corresponding message in a pop-up window in the bottom-left corner of the editor.

Tip The authentication dialog appears immediately only when you attempt to access a URL address outside your working copies. This feature applies to Perforce, Subversion, and CVS.

To access the authentication dialog box

1. Click the Notifications Pending button on the Status bar.

2. In the Notifications pop-up window that opens, click the link Click to fix.



3. In the authentication dialog box that opens, specify your credentials.

VCS-Specific Procedures

In most cases, PhpStorm provides a unified approach to version control operations, as described in the previous sections. Nevertheless, there are certain VCS-specific features and peculiarities that the user should be aware of. Find helpful tips and notes in the following sections:

- Using CVS Integration
- Using Git Integration
- Using Perforce Integration
- Using Mercurial Integration
- Using Subversion Integration
- Using TFS Integration
- Using GitHub Integration

Using CVS Integration

With the CVS integration enabled, you can perform basic CVS operations from inside PhpStorm.

Note The information provided in the topics listed below assumes that you are familiar with the basics of CVS version control system.

In this section:

- Browsing CVS Repository
- Checking Out Files from CVS Repository
- Configuring CVS Roots
- Configuring Global CVS Settings
- Ignoring Files
- Importing a Local Directory to CVS Repository
- Resolving Commit Errors
- Updating Local Information in CVS
- Using CVS Watches
- Working Offline
- Working with Tags and Branches

Prerequisites

- PhpStorm comes bundled with the CVS plugin. This plugin is turned on by default. If it is not, make sure that the plugin is enabled.
- PhpStorm CVS integration does not require a standalone CVS client. All you need is an account in your CVS repository.
- CVS integration is enabled for the current project root or directory.

CVS support

- When CVS integration with PhpStorm is enabled, the CVS item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the CVS control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using CVS integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated CVS commands.
  - Error messages.

Browsing CVS Repository

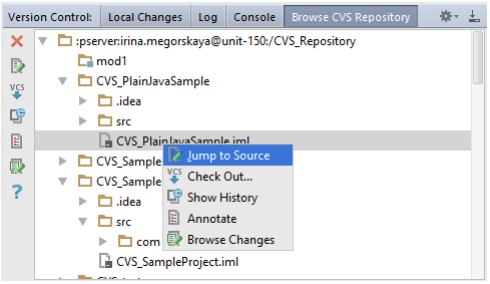
You can browse any CVS repository and modify the structure of the currently open project, or a different one. Browsing contents of a repository is always available, even when CVS is not enabled in project. All you need is a valid user account.

To browse the CVS repository and modify its structure

1. Open a project. Then, choose VCS | Browse VCS Repository | Browse CVS Repository... on the main menu. The Select CVS Root Configuration dialog is opened.

2. Select a root from the list of configured CVS roots, or click Configure to specify a new one , and then click OK. The Browse CVS Repository tab opens in the CVS tool window at the bottom of the Editor.

3. Browse the desired CVS repository and perform jump to source, checkout, browse changes and annotate operations for files and folders.



Icons for the tree nodes denote their respective types. For example:

– means a CVS directory.

– denotes a CVS module.

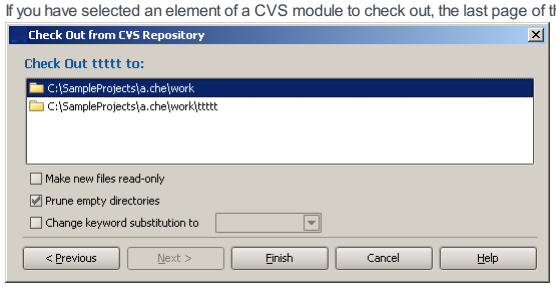
Icons appearing next to the files will denote the corresponding file types.

Checking Out Files from CVS Repository

Checking out action helps you obtain a writable copy of the repository, which you can edit as required. After making the necessary changes, you can publish results by committing, or checking in, to the repository. This section describes CVS-specific checkout procedure.

To check out files from a CVS repository

1. On the main menu, choose VCS | Checkout from Version Control.
2. On the submenu, choose CVS.
3. In the Checkout From CVS Repository dialog , select the desired CVS configuration, and click Next. If the necessary CVS configuration is not available, click Configure to create a new one .
4. Select the elements to check out, and click Next.
5. Specify the checkout destination directory where the local copy of the repository files will be created and click Next.
   If you are checking out sources for an existing project, the destination folder should be below a project content root .
6. If you have selected an element of a CVS module to check out, the last page of the wizard suggests you to add the module name to the local path:



7. Set up CVS checkout options, or accept defaults, and click Finish.
8. PhpStorm suggests to create a project based on the sources, checked out from version control.
   If you accept the suggestion, open new project, as described in the section Opening Multiple Projects .

Configuring CVS Roots

CVS roots are global PhpStorm properties. Once configured, a CVS root is available regardless of which project is currently opened. This is helpful if you need to check out an entire project from CVS.

You can define multiple CVS root configurations for future use and edit them whenever necessary. Alternatively, you can configure CVS roots when checking out files or directories from or importing them to a CVS repository.

From this section you will learn how to:

- Configure a new CVS root
- Modify a CVS root
- Configure a CVS root based on an existing configuration
- Remove a CVS root configuration

To configure a CVS root, follow these general steps:

1. Open the CVS Roots dialog box.
2. Click the Add button + on the toolbar.
3. Specify the CVS root string .
4. Specify the version to work with.
5. Specify additional connection settings .
6. Click the Test connection button to check that the specified settings ensure establishing successful connection to the CVS server.
7. Click OK to apply the specified settings and close the dialog box.

To modify an existing CVS root configuration

1. Open the CVS Roots dialog box.
2. Select the CVS root to modify.
3. Edit the settings as while configuring a new CVS root.
   Changes made here apply to the current CVS root configuration only.

To configure a new CVS root based on an existing configuration

1. Open the CVS Roots dialog box.
2. Select the CVS root to be used as the basis for a new configuration.
3. Click the Copy button on the toolbar or press `Ctrl+O`. The selected configuration is copied as a new CVS root.
4. Edit the newly created CVS root configuration, as required.

To remove a CVS root configuration

1. Open the CVS Roots dialog box.
2. Select the CVS root you want to remove and click the Remove button — on the toolbar.

Accessing the CVS Roots Dialog Box

You can access the CVS Roots dialog box in several ways, depending on the general task you are currently performing.

To open the CVS Roots dialog box, do one of the following:

- When checking out files or directories from a CVS repository: on the main menu, choose VCS | Checkout from Version Control | CVS and in the dialog box that opens click the Configure button.
- When importing files or directories to a CVS repository: on the main menu, choose VCS | Import into CVS and in the dialog box that opens click the Configure button.
- When defining a CVS root to use in the future: open a file which is already under CVS control and choose VCS | CVS | Configure CVS Roots on the main menu.

Assembling a CVS Root String

CVS root strings are specified in the CVS Root text box of the CVS Roots dialog box.

The CVS Root string syntax is:

`[:method:][[user][:password]@]hostname[:[port]]/path/to/repository`.

You can obtain the valid string from your system administrator, or assemble the CVS root parameters into a correct string manually, or use the Edit by Field functionality, as described below.

To assemble the CVS Root parameters

1. In the CVS Roots dialog box, click the Edit by Field button next to the CVS Root text box. The Configure CVS Root Field by Field dialog box opens.
2. Choose the connection method and specify the user name, port, host, and repository. See the Configure CVS Root Field by Field dialog box reference for details.
3. Click OK. The dialog box closes and you return to the CVS Roots dialog box. The CVS Root text box displays the specified parameters assembled in a valid string.

Specifying a Version to Work With

By default, PhpStorm suggests you to check out the latest (HEAD) revision to work with. However, you can synchronize your local working copy with any previous revision from the repository.

Tip Make sure that the CVS root field is filled in with valid data.

To specify the version to work with

1. In the Use Version section of the CVS Roots dialog box, select the criterion to search for the desired version. The available options are:
   - By tag
   - By date

2. Click the Browse button ... next to the selected option and do one of the following depending on the selected search criterion:
   - If you have selected By tag, select the desired tag from the list of tags that opens. The list displays the tags obtained from the CVS server according to the specified CVS root string.
   - If you have selected By date, select the date and time in the calendar that opens.

   You can also type the desired revision number, tag, or date in the text box next to the selected option.

Specifying Additional Connection Settings

You can flexibly configure connection to the CVS server using additional settings. The set of relevant options depends on the connection method specified in the CVS root string. The available additional options for each of the supported connection methods are displayed in the lower part of the CVS Roots dialog box. See the CVS Roots dialog box reference for details.

Configuring Global CVS Settings

For a module associated with CVS you can specify global CVS settings, which includes character set, location of the password file, connection timeout etc.

To configure CVS global settings for a directory associated with CVS

1. On the main menu, choose VCS | CVS | Global Settings.

2. In the Global CVS Settings dialog box that opens, specify the global settings and click OK.

Tip Alternatively, you can define global CVS settings when configuring a CVS root . To invoke the Global CVS Settings dialog box, click the Global Settings button in the CVS Roots dialog box.

Ignoring Files

If you want your CVS integration to ignore certain unversioned files under CVS-associated directories, and skip them when performing update, import etc., add these files to the CVS ignore list, which is stored in the `.cvsignore` file.

The way CVS integration handles unversioned files depends on the general settings for file creation . If the new files, created with PhpStorm, are not put under version control automatically, you can add them to the ignore list using CVS command. You can put the `.cvsignore` file under CVS version control, and it will be recognized by all CVS clients.

To include an unversioned file to the ignore list

1. Select an unversioned file under a CVS-associated directory (by default, such files are brown).
2. On the main VCS menu, or on the context menu of the selection, choose CVS | Ignore. If `.cvsignore` file is not under CVS version control, proceed to the next step. If `.cvsignore` already exists and is under version control, PhpStorm adds the file in question to the ignore list silently.
3. If you want to put the ignore list under version control, in the Add File .cvsignore to CVS dialog box, click Add to CVS and optionally specify the desired keyword substitution .

Tip If you want to remove a file from the ignore list, you can only do it manually. Open the `.cvsignore` file for editing by pressing `F4`, and remove the lines for the files that should not be ignored by CVS.

Importing a Local Directory to CVS Repository

You can import an entire directory to your CVS repository, provided that you have the corresponding access rights. This action is helpful for putting a whole project under version control.

Import to the repository is always available, even when CVS is not enabled in project.

To import a directory into CVS repository

1. On the main menu, choose VCS | Import into CVS.
2. On the first page of the Import Into CVS wizard, select the target CVS root. If the desired target repository does not exist, you can create a new one. To do that, click Configure, and define the desired CVS root . Click Next.
3. On the second page of the wizard, select the target directory in the repository, and click Next.
4. On the third page of the wizard, select the local directory that will be imported into the CVS repository. Click Next.
5. In the Customize Keyword Substitution page, specify the keyword substitution rule for the files imported into the repository, and click Next.
6. Specify the required import settings . The fields of this page correspond to CVS command-line arguments for *import*, and additional options that define the checkout status of the imported directory.
7. Click Finish.

Resolving Commit Errors

In the section Checking in Files , you have learnt how to check in (commit) your changes to the repository. In this section you can find examples of CVS-specific error messages and suggestions on resolving conflicts.

If any error occurs when trying to commit, PhpStorm displays an error message. For example:

- If you have a modified file, which has been already changed on the server by someone else, since your last synchronization, you will get the following error:

```
Error: cvs server: Up-to-date check failed for 'source/com/...'
```

  In this case you would need to merge your local copy with the current revision in the repository. When the copies are merged, and all possible conflicts are resolved , so that the file is assigned the merged status, you can safely commit it to the repository.

- If you try to commit a file marked with a sticky tag, or sticky date, the CVS server will detect an attempt to *change the past*, and the error looks as follows:

```
Error: cvs server: sticky tag '1.1' for file 'source/com/impl/ManagerImpl.java& is not a branch
```

  To solve the problem, you need to update with resetting sticky data; in this case your changes will be merged with the most recent revision of the file. After resolving possible conflicts (by calling the Merge command) you will be able to commit the files.

Updating Local Information in CVS

From the Updating Local Information topic, you have learnt the general procedure. CVS integration provides a special Update File / Directory dialog box with the options that map directly to the corresponding CVS command-line options of the `update` command. Refer to the CVS documentation for details.

This section will consider the CVS-specific procedure and several of the options in terms of their presentation in PhpStorm.

To update local information

1. Select one or more files and/or directories in any navigation view (for example, Project tool window).

2. On the main menu, select VCS | Update Project or on the context menu of the selection choose CVS | Update File (Directory). The Update dialog for a project or file opens. In case of project update, select CVS tab.

3. Specify the following options:
   Branch Merging

   You can choose to merge your local file(s) with the counterpart(s) in one or two CVS branches. In the CVS command-line interface, this is the `-j` option.

   The option *Don't Merge* is selected by default, as merging across branches is not commonly needed. If you choose one of the other options, one or both of the text fields are enabled (depending on the choice of options). Clicking the Browse button next to each field opens the Select Tag dialog box which lists all the branch tags maintained by the repository on the CVS server. Locate the branch you want to merge with, select it in the list, and click OK.

   Use Version

   You can optionally update your local system from some different revision. You can choose a revision by its tag or by its date. The Default option synchronizes with your file's current revision, as this is the most common synchronization. The other options for Use Version are:
   - By tag (-r):When updating a single file, you can choose the revision either by Revision or Tag. When you choose this option, the text field and corresponding ellipsis button are enabled. Enter a revision number or tag in the text field, or click the ellipsis button and select either a revision or a tag in the resulting dialog.

     When updating multiple files (selected individually, or when invoking update on a directory), you can only select the revision by Tag.
   - By date (-d): You can update from the revision of a specific date. This is possible whether you are invoking update on one file or multiple files or directories. When you choose this option, the date defaults to the current date. To specify a different date, click the ellipsis button and specify the desired date in the Choose Date dialog which appears.

   Reset sticky data (-A)

   If the last checkout or update of the selected file(s) was from some revision that was specified by tag or date, the tag/date information is *sticky* for the file(s). If you now want to update these sticky-tagged files from the *HEAD* revision, select this option in combination with selecting the Default option in *Use Version* so that the sticky information is removed.

   Change keyword substitution to

   If checked, this is converted to the -k CVS parameter. Refer to CVS Options: Default keyword substitution for text files for details.

   Do not show this dialog in the future

   Select this option if you want the update operation take place silently. For details see CVS Options.

   To have PhpStorm show this dialog box before update again:
   1. Open the Version Control - Confirmation page of the Settings dialog box.
   2. In the Display Option dialogs when these commands are invoked area, select the Update check box.

Using CVS Watches

CVS integration of PhpStorm helps coordinate the activities of team members, who concurrently work on the same files or directories.

CVS watches enable you to notify the users of a CVS repository whenever a file has been opened for editing, or committed. If you watch a file or directory for changes and commits, you are added to the list of watchers.

Usage the commands related to watches depends on configuration of the `$CVSROOT/CVSROOT/notify` file.

Edit and Unedit commands change read-only status of the files or directories under CVS. If the sources were checked out with the option `-c`, you can apply Edit command to make them writable. In this case you are added to the list of editors. When you are done with editing, use Unedit command to restore read-only status. So doing, you are removed from the list of editors. If watching is configured, the watchers will receive email notification about these events.

In fact, the option Use read-only flag for not edited files in the CVS provides the same functionality automatically. If this option is checked, Unedit always applies to the source files after commit.

Edit and Watch commands apply to all the files you have selected in the current view (including all the files in any selected directory), or to the current file in the editor if you invoked the command there.

This section describes how to:

- Access Edit and Watch commands
- Change read-only status of a file or directory
- View the other persons who edit the same file or directory
- Set watch on a certain event for a file or directory and thus add yourself to the list of wanchers
- Enable or disable watching
- View the other persons who watch the same file or directory

To access Edit and Watch commands

1. In one of the tool windows, select the desired files or directories, or open a file in the editor.
2. Do one of the following:
   - On the main menu, choose VCS | CVS | Edit and Watch
   - On the context menu, choose CVS | Edit and Watch

To get write access to a file or directory

1. Open Edit and Watch menu .
2. Choose Edit on the submenu. Edit Options dialog box is displayed.
3. If you want to gain exclusive write access, check the option Reserved edit (-c). Click OK.

To restore read-only status of a file or directory

1. Open Edit and Watch menu .
2. Choose Unedit on the submenu.

To view the other persons who edit the same file or directory

1. Open Edit and Watch menu .
2. Choose Show Editors on the submenu. This will display the list of all users who have run the Edit command on the same file or directory.

To set watch on a file or directory

1. Open Edit and Watch menu .
2. Choose Add Watch on the submenu.
3. In the dialog box that opens, select the type of action you would like to be notified about:
   - Edit: you will notified whenever Edit is applied to a watched file or directory.
   - Unedit: you will notified whenever Unedit is applied to a watched file or directory.
   - Commit: you will notified whenever Commit is applied to a watched file or directory.
   - All: you will notified whenever any of the above commands is applied to a watched file or directory.

To remove watch from a file or directory

1. Open Edit and Watch menu .
2. Choose Remove Watch on the submenu.
3. In the dialog box that opens, select the type of action for which you would like to skip notification (Edit, Unedit, Commit or All).

To suspend or resume watching

1. Open Edit and Watch menu .

2. Choose Watch Off or Watch On on the submenu.

To view the list of users who are watching the same files or directories

1. Open Edit and Watch menu .

2. Choose Show Watchers on the submenu.

Working Offline

Offline mode in CVS makes it possible to ignore network errors. When this mode is enabled, you will not receive any notifications about connection problems.

You can go to offline mode in two ways

- When a connection error occurs, PhpStorm informs you about that. With the first successful transaction, offline modes automatically turns off.
- Using the CVS menu command on the main Version Control menu, or a context menu: VCS | CVS | Work Offline.

Working with Tags and Branches

From within PhpStorm, you can create and delete CVS tags and branches. The names of the tags and branches must start with a letter, and contain only alphanumeric characters.

Branch and tag commands apply to all the files you have selected in the current view (including all the files in any selected directory), or to the current file in the editor if you invoked the command there.

This section describes how to:

- Access the tags and branches commands .
- Create a branch in the repository on the base of the current revision.
- Tag a revision in you local working directory .
- Delete a tag .

To access tags and branches commands

1. In one of the tool windows, select the desired files or directories, or open a file in the editor.
2. Do one of the following:
   - On the main menu, choose VCS | CVS
   - On the context menu, choose CVS
3. On the submenu, choose the appropriate command (Create Branch, Create Tag, or Delete Tag)

To create a branch

1. Invoke the Create Branch command.
2. In the Create Branch dialog box, specify the name of the new branch. To do that, type the name in the Branch name field, or click the Browse button ... and select the desired name from the list of existing CVS tags.
3. Optionally, specify the following:
   - Select the Override existing check box, if you want to move an existing tag to a new branch, as defined by the option `-F` of `rtag` CVS command.
   - Select the Switch to this branch check box to switch your local working copy to the branch specified in the Branch name field.

To create a new tag

1. Invoke the Create Tag command.
2. In the Create Tag dialog box, specify the new tag name. To do that, type the name in the Tag name field, or click the Browse button ... and select the desired name from the list of CVS tags.
3. Optionally, specify the following:
   - Select the Override existing check box, if you want an existing tag to point to the current revision, as defined by the option `-F` of `rtag` CVS command.
   - Select the Switch to this tag check box to switch your local working copy to the tag specified in the Tag name field.

To delete a tag

1. Invoke the Delete Tag command.
2. In the Delete Tag dialog box, specify the name of the tag you want to delete. To do that, type the name in the Tag name field, or click the Browse button ..., and select tag name from the list of the current tags in the repository.

Using Git Integration

With the Git integration enabled, you can perform basic Git operations from inside PhpStorm.

> **Note** The information provided in the topics listed below assumes that you are familiar with the basics of Git version control system.

In this section:

- Adding Files to a Local Git Repository
- Adding Tags
- Checking Git Project Status
- Committing Changes to a Local Git Repository
- Fetching Changes from a Remote Git Repository
- Handling Passwords for Git Remote Repositories
- Managing Branches
- Setting up a Local Git Repository
- Stashing and Unstashing Changes
- Pulling Changes from the Upstream (Git Pull)
- Pushing Changes to the Upstream (Git Push)
- Handling LF and CRLF Line Endings

Prerequisites

- Git is installed on your computer.
  It is strongly recommended that you use version 1.7.1.1 or higher.
- The location of the Git executable file is correctly specified on the Git page of the Settings dialog box.
- Git integration is enabled for the current project root or directory.
- If you are going to use a remote repository, create a Git hosting account first. You can access the remote repository through the username/password and keyboard interactive authentication methods supported by the Git integration or through a pair of `ssh` keys.
  Please note the following:
  1. `ssh` keys are generated outside PhpStorm. You can follow the instructions from https://help.github.com/articles/generating-ssh-keys/ or look for other guidelines.
  2. Store the `ssh` keys in the home_directory `\.ssh\` folder. The location of the home directory is defined through environmental variables :
     - `HOME` for Unix-like operating systems.
     - `userprofile` for the Microsoft Windows operating system.
  3. Make sure, the keys are stored in files with correct names:
     - `id_rsa` for the private key.
     - `id_rsa.pub` for the public key.
  4. PhpStorm supports a standard method of using multiple `ssh` keys, by means of creating `.ssh/config` file.

Git support

- When Git integration with PhpStorm is enabled, the Git item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the Git control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using Git integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated Git commands.
  - Error messages.

Adding Files to a Local Git Repository

On this page:

- Basics
- Adding all currently unversioned files to Git control
- Adding specific files to a local Git repository

Basics

After a Git repository for a project is initialized , you need to add the project data to it.

If you have specified Git as the version control system for your project in the Settings | Version Control page, PhpStorm suggests to put each newly created file under Git version control.

To make Git ignore some types of files, configure files to ignore .

Adding all currently unversioned files to Git control

1. Switch to the Version Control tool window and switch to the Local Changes tab. tool window.
2. Expand the Unversioned Files node and choose Add to VCS from the context menu or press `Ctrl+Alt+A`.

Adding specific files to a local Git repository

Do one of the following:

- Switch to the Version Control tool window and switch to the Local Changes tab. Expand the Unversioned Files node, and select the files to be added. From the context menu, choose Add to VCS, or press `Ctrl+Alt+A`.
- Switch to the Project tool window and select the files to be added. From the context menu, choose Git | Add or press `Ctrl+Alt+A`.

Adding Tags

On this page:

- Basics
- Assigning tags to commits
- Creating annotated tags

Basics

The Git integration supports tagging a particular commit or object to refer to it in the future. Basically, by adding a tag you create a branch that never moves.

You can also create annotated tags and have annotations displayed to facilitate viewing and navigating through tagged commits.

Assigning tags to commits

To assign a tag to a commit

1. On the main menu, choose VCS | Git | Tag Files. The Tag dialog box opens.

2. From the Git Root drop-down list, select the required local repository.
   The Current Branch read-only field shows the branch you are currently working with in the selected repository. The contents of the read-only field change as you change the selection in the Git Root drop-down list.

3. In the Tag Name text box, type the name of the new tag.

   Tip If you specify a name that already exists, PhpStorm displays an error message. To override the error and re-assign the tag, select the Force check box.

4. In the Commit text box, specify the commit or object which you want to tag:
   - To tag the latest commit in the branch (HEAD), leave the text box empty.
   - To tag a particular commit, specify its commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`.
     Refer to the Git commit naming conventions for details.

5. To check that the specified commit exists and is the one you actually need, click the Validate button. The Paths affected in commit dialog box shows the files that were affected in the specified commit. View the information and click OK.
   If you specify a commit that does not exist, PhpStorm displays an error message.

Creating annotated tags

To create an annotated tag

1. Create a tag.

2. In the Message text box, provide a description of the commit.

Checking Git Project Status

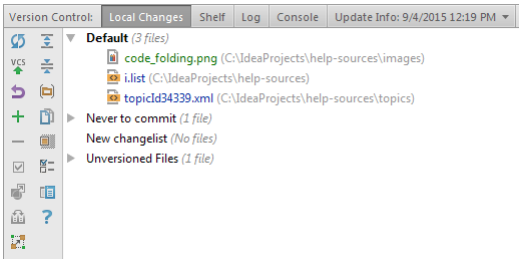
Besides indicating the current file status relative to the repository, PhpStorm integration with Git provides you with the accumulated view of the project files' statuses.

To view the differences between the current state of project files and the repository

1. Open the required project.

2. On the main menu, choose VCS | Refresh File Status.

3. Switch to the Version Control tool window and open the Local Changes tab.



The status of each file is indicated by the color in which the path to the file is displayed.

Committing Changes to a Local Git Repository

This topic describes how to commit changes to a local Git repository . For information on uploading changes to a remote repository, see Pushing Changes to the Upstream (Git Push) .

To commit changes to a local repository

1. Open the Version Control tool window by pressing `Alt+9` or choosing View | Tool Windows | Version Control.

2. In the Local Changes tab , select the files or folders you want to commit.

3. Open the Commit Changes dialog box by doing one of the following:
   - On the tool window toolbar, click the Commit Changes button .
   - On the main menu, choose VCS | Commit Changes.
   - On the main menu, choose VCS | Git | Commit File.
   - Press `Ctrl+K`.

4. The Commit Changes dialog lists all files that have changed since the last commit. Select the check boxes next to the files you want to commit now.

5. Add a comment in the Commit Message section and select the Before Commit options you want PhpStorm to perform on the files before committing them to the local repository.

   Tip To upload the committed changes to the remote repository immediately, select the Push Changes check box. The changes will be pushed to the pre-configured origin.

6. In the Git section, configure the following settings:
   - Use the Author drop-down list to specify the person who created the changes to be committed.

     Tip This may be necessary when you are committing changes made by another person.

   - If you want to replace already published commit, select the check box Amend commit, see Git Basics: Undoing Things for details.

7. Click Commit.

   Tip To save the changes in a text file, click the Create Patch button.

Fetching Changes from a Remote Git Repository

The Fetch operation involves downloading changes from a remote repository without applying them locally.

Tip Basically, the Fetch operation is intended for downloading changes from a remote repository that is different from the Origin .

To fetch changes from a remote repository

- On the main menu, choose VCS | Git | Fetch .
  PhpStorm retrieves the changes from the repository silently.

Handling Passwords for Git Remote Repositories

On this page:

- Basics
- Configuring the password policy
- Setting the master password
- Changing the master password
- Resetting the master password

Basics

Every time you interact with a remote Git repository, for example, during Pull, Update, or Push operations, you are requested to specify the private `ssh` key or passphrase to identify yourself. You may happen to use a number of remote repositories and accordingly need to remember a number of passwords.

PhpStorm provides the ability to configure a password policy according to which passwords are either never saved, or saved during one session and cleared upon the session end, or stored in a special passwords database.

The passwords database is under protection of a master password. Once specified , the master password can be changed or reset at any time. PhpStorm stores the history of all updates to the master password until it is reset.

Warning! When you reset the master password, the history of all the previously used master passwords is removed.

Configuring the password policy

To configure the password policy

1. Open the IDE Settings , then click Passwords.
2. On the Passwords page that opens, specify how you want PhpStorm to process passwords for Git remote repositories.
   Do one fo the following:
   - If you do not need PhpStorm to save passwords at all, select the Do not remember passwords option.
   - To have passwords stored in the memory during a session, select the Remember passwords until closing of the application option.
   - To have passwords stored in a passwords database, select the Remember on disk (protected with master password) option and specify the master password for the storage.

Setting the master password

To set the master password

1. On the Passwords page of the Settings dialog box, select the Remember on disk (protected with master password) option.
2. Click the Reset button.
3. In the Master Password dialog box that opens, specify the password to use. Type the desired password once again to confirm your setting.

Changing the master password

To change the master password

1. On the Passwords page of the Settings dialog box, click the Change Password button.
2. In the Change Master Password dialog box that opens, type the currently used master password and the password to change it to. Confirm the new password by typing it once again.

Resetting the master password

To reset the master password

1. Open the Reset Master Password dialog box. Do one of the following:
   - On the Passwords page of the Settings dialog box, click the Reset Master Password button.
   - In the Change Master Password dialog box, click the Reset Password button.
2. In the New Password text box, type the master password to replace the current one with.
3. In the Confirm Password text box, type the new master password once again to confirm your setting.

Managing Branches

PhpStorm supports processing of multiple Git branches using sophisticated checkout and merge strategies, and provides an interface for configuring operations on branches.

To manage branches, use the Branches node and the nested commands. This node is available on:

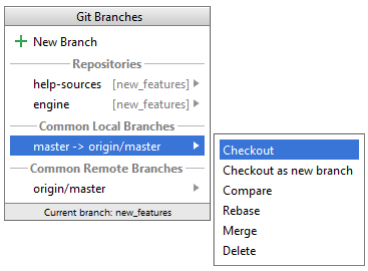
- the main VCS menu
- context menu of the editor
- VCS operations pop-up

In this part:

- Accessing Git Branches Popup Menu
- Creating a New Branch
- Merging, Deleting, and Comparing Branches
- Rebasing Branches
- Checking Out (Switching Between) Branches
- Resetting Branch Head to a Selected Commit
- Resetting Head Commit
- Applying Changes from a Specific Commit to Other Branches (Cherry Picking)
- Git Branches in Multirooted Projects

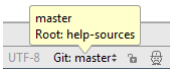
Accessing Git Branches Popup Menu

The Git Branches popup menu contains all commands that are required to manage Git branches, and can be invoked in a number of ways.



To open the Git Branches popup menu, do one of the following:

- From the main menu, choose VCS | Git | Branches.
- Right-click any file in the editor, and in the context menu choose Git | Repository | Branches.
- In the VCS operations popup menu, choose Branches.
- Click the Git widget in the Status bar:



Creating a New Branch

On this page:

- To create a new Git branch
- To check out a new Git branch from a local branch
- To check out a new local branch from a remote branch

To create a new Git branch

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. In the pop-up menu, choose New Branch.
3. In the Create new branch dialog box, specify the branch name. The branch with the specified name will be checked out (corresponds to `git checkout -b`).

To check out a new Git branch from a local branch

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Checkout as new branch from the submenu.
3. Specify the branch name in the Create new branch dialog that opens.

To check out a new local branch from a remote branch

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Checkout as new local branch from the submenu.
3. Specify the name of the new branch in the Checkout new branch from <branch name> dialog that opens.
   The branch with the specified name will be checked out and put under version control.

Merging, Deleting, and Comparing Branches

On this page:

- To merge a branch
- To delete a branch
- To compare branches

To merge a branch

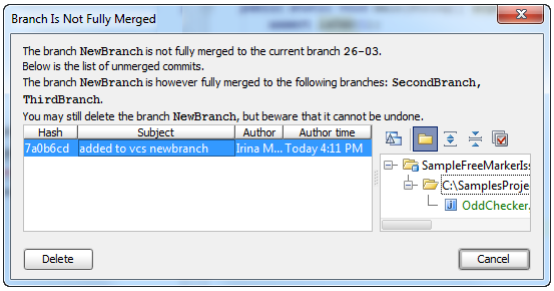
1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Merge from the submenu. The selected branch will be merged into the branch that is currently checked out .
   If there are merge conflicts, you will be prompted to resolve them.

   If your local changes are going to be overwritten by merge, PhpStorm suggests to perform smart merge (similar to smart checkout .

To delete a branch

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Delete from the submenu.
   If there are no unmerged commits, the selected branch will be deleted.

   If there are commits that have not yet been merged into the selected branch, a dialog will open with a list of unmerged changes, and the list of branches where the selected branch has already been merged into. This information should help you decide if you still want to delete the selected branch.



   If you are sure you want to delete the selected branch, click Delete and confirm the operation.

To compare branches

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Compare from the submenu.
   PhpStorm compares the branch that is currently checked out with the selected branch.
3. In the dialog that opens, compare the differences in the following two tabs:
   - Log: this tab lists the commits that exist in the current branch, and are missing in the selected branch, and vice versa.
   - Diff: this tab shows the differences between files existing in both branches. Use the Show Diff command on the context menu of a file to explore the differences between branches.

Rebasing Branches

Git integration with PhpStorm supports the Rebase operation and provides an interface that ensures high flexibility in setting the rebase arguments.

The following functionality is supported:

- The basic use case which involves applying a branch on top of the current HEAD of the master after synchronization with the upstream.
- Rebasing a branch entirely or partially to a specific commit in any branch or tag.
- Running rebase on several local repositories simultaneously.
- Selecting a merge strategy to apply, with the possibility to use no merging strategy at all.
- Running rebase interactively with control over preserving/squashing merges.
- Resuming interrupted rebase after merge conflicts are resolved.
- Cancelling rebase.

To initiate a rebase operation

1. From the main menu, select VCS | Git | Rebase. The Rebase Branches dialog box opens.
2. From the Git Root drop-down list, select the relevant local repository.
3. From the Branch drop-down list, select the branch you want to rebase. By default, the current branch is selected. If you specify a different branch, it will be checked out.
4. Specify the new base and the commits you want to apply.
5. If necessary, choose a rebase strategy and click Rebase.

The `rebase` command is also available from the Git Branches popup in the submenu for the selected branch.

To resume an interrupted rebase operation

- From the main menu, choose VCS | Git | Continue Rebasing.
  Before resuming the rebase operation, view the log in the Version Control tool window.

If rebase was initiated and interrupted on two or more local repositories, the Continue Rebasing dialog box is displayed. Select the repository on which you want to resume the rebase operation from the Git Root drop-down list.

To cancel a rebase operation

- From the main menu, choose VCS | Git | Abort Rebasing

If rebase was initiated on two or more local repositories, the Abort Rebasing dialog box is displayed. Select the repository on which you want to cancel the rebase operation from the Git Root drop-down list.

Applying a Branch Entirely on Top of Master

Suppose you have `branch1` based on `master`. While you were working in `branch1`, some updates were committed to `master`.

The diagram below illustrates rebasing `branch1` so that it applies on top of the current HEAD of `master`.

By default, commits `1`, `2`, and `3` are applied one after another in the chronological order. To skip, edit, squash commits or change their order, run the rebase operation in the interactive mode .

To apply a branch entirely on top of the current HEAD of the master

1. Initiate the rebase procedure.
2. Clear the Preserve Merges check box.
3. Clear the Interactive check box.
4. From the Onto drop-down list, select the master branch.
5. Clear the selection in the From drop-down list, if anything is selected.
6. From the Merge Strategy drop-down list, select Default.
7. Click the Rebase button. The rebase process starts. View the rebase log in the Version Control tool window, resolve conflicts that arise, and resume rebasing.

Interactive Rebase

Rebasing branches interactively provides you with the possibility to apply commits in the necessary order, squash or edit commits before applying, and skip commits that contain extraneous changes.

Interactive mode is available for rebasing branches entirely or partially on top of the HEAD or to any specific commit.

To rebase a branch in the interactive mode

1. Initiate the rebase procedure.
2. Select the Interactive check box.

   Tip To have PhpStorm try to recreate merges instead of ignoring them, select the Preserve Merges check box.

   Warning! Git does not support squashing commits when the Preserve merges option in enabled.
3. Specify the new base , the range of commits to apply, and the merge strategy .
4. Click OK. The Rebasing Commits dialog box opens displaying a list of all the commits within the specified range in the chronological order. For each commit its hash and comment are shown.

   Tip To view which files are affected in a commit, select the commit and click the View button.
5. Define the order of processing the commits by selecting the relevant lines and clicking the Move Up and Move Down buttons.
6. Use the Action drop-down list to define how each commit should be processed:
   - To apply a commit as is, select the Pick option.
   - To update a commit before applying, select the Edit option.
   - To ignore a commit, select the Skip option.
   - To combine a commit with the previous commit, select the Squash option.
     After you start rebasing, you will be asked to supply additional information on the squashed commits.

     If the affected commits have different authors, the squashed commit will be attributed to the author of the first commit.
7. Click the Rebase button. The rebase process starts. View the rebase log in the Version Control tool window, resolve conflicts that arise, and resume rebasing.

Rebasing a Branch to a Specific Commit

Suppose you have `branch1` based on `master`. While you were working in `branch1`, some updates were committed to `master`. For some reason, you do not want to rebase `branch1` to the current HEAD of `master`, but want to rebase it to commit `CommitB`. Moreover, you do not want to apply commit `1`, but want to start with commit `2`.

The diagram below illustrates the described rebase operation.

Transplanting a branch to a branch different from its base is also supported. Suppose your `branch2` is based on `branch1`, which, in its turn, is based on `master`.

The diagram below illustrates rebasing `branch2` to a specific commit in `master`.

To rebase a branch to a specific commit

1. Initiate the rebase procedure.

2. Configure the rebase mode:
   - To interactively handle the list of commits to apply, select the Interactive check box.
   - To have PhpStorm try to recreate merges instead of ignoring them, select the Preserve Merges check box.
     Note that this option is available only in the interactive mode.

3. Define the contents of the Onto and From drop-down lists. By default, tags and remote branches are not available from them. To expand the lists with tags or remote branches, select the Show Tags and Show Remote Branches check boxes respectively.

4. From the Onto drop-down list, select the required branch and specify the commit to move the base to. Type the desired commit hash or use an expression, for example, of the following structure:

   ```
   <branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>
   ```

   Refer to the Git commit naming conventions for details.

   > **Tip** If you select a tag, no commit specification is required.

5. Click the Validate button and view view which files are affected in the specified commit in the Paths affected in commit... dialog box that opens.

6. From the From drop-down list, select the current branch and specify the commit, starting from which you want to apply commits on top of the new base. Type the desired commit hash or use an expression, for example, of the following structure:

   ```
   <branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>
   ```

   Refer to the Git commit naming conventions for details.

   > **Tip** To apply all commits, leave the field empty.

7. Specify the merge procedure to be applied, when necessary:
   - To perform all merges manually, select the Do not use merging strategies check box.
   - To have PhpStorm apply one of the available merging methods, select the relevant option from the Merge Strategy drop-down list. See the Rebase Branches dialog box reference for description of available options.

8. Click the Rebase button and proceed depending on the chosen rebase mode:
   - If you have specified the interactive mode, configure the list of commits to apply in the Rebasing Commits dialog box, that opens.
   - If you have specified the automatic mode, view the rebase log in the Version Control tool window, resolve conflicts that arise, and resume rebasing

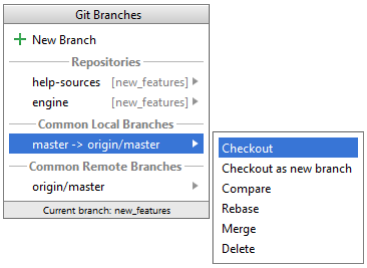
Checking out (Switching Between) Branches

On this page:

- Checking out an existing branch
- Checking which branch is currently checked out

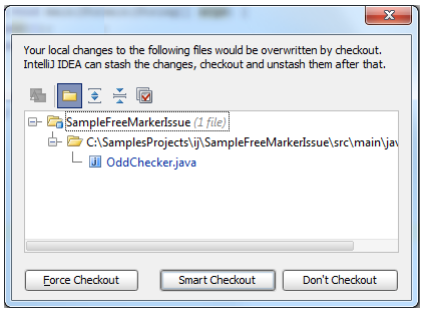
Checking out an existing branch

1. Invoke the Branches menu as described in Accessing Git Branches Popup Menu .
2. Select a branch in the pop-up list that shows all available local and remote branches, and choose Checkout from the submenu.



What happens next depends on whether there are conflicts:

- If your working tree is clean and your local changes do not conflict with the specified branch, this branch will be checked out (a notification will popup in the bottom-left corner of the PhpStorm window).
- If you ware working with a dirty tree, and your local changes will be overwritten by checkout, PhpStorm shows the files that prevent you from checking out the selected branch, and suggests to choose between the `force checkout` and `smart checkout` .



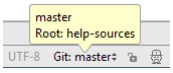
If you select `force checkout` , your local changes will be overwritten (equivalent to the `git checkout -f` command).

If you select `smart checkout` , PhpStorm will stash local changes, check out the selected branch, and then unstash the changes. If a conflict occurs during the unstash operation, you will be prompted to merge the changes.

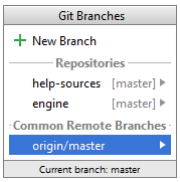
Checking which branch is currently checked out

If you want to check which branch is you are currently working in, do one of the following:

- Look ath the widget in the Status bar :



- Invoke the Branches menu as described in Accessing Git Branches Popup Menu . . Check the current branch name in the bottom line:



Resetting Branch Head to a Selected Commit

PhpStorm allows you to reset the current Git branch head to a selected commit and update the working tree and the index.

To reset the current branch head to a selected commit

1. In the main menu navigate to View | Tool Windows | Changes.
2. In the Version Control tool window switch to the Log tab .
3. Select a commit that you want to reset the current branch head to, and right-click this commit. Select Reset Current Branch to Here from the popup menu.
4. In the Git Reset dialog that opens, select the mode for updating the working tree and the index. The following options are available:
   - Soft: The modified files will not be changed, and the differences will be staged for commit.
   - Mixed: The modified files will not be changed, and the differences will be unstaged. This mode is used by default and is identical to the `git reset` command.
   - Hard: The modified files will be reverted to the selected commit, and all uncommited changes will be lost.
   - Keep: The modified files will be reverted to the selected commit, but local changes will be preserved.

Resetting Head Commit

Suppose, you notice a small error in a recent commit or a set of commits and want to redo that part without showing the undo in the history. In this case resetting the latest commit in the current branch (HEAD) to an earlier commit is helpful. The Reset HEAD operation sets the current HEAD to the specified commit and optionally resets the index and working tree to match.

To reset the current HEAD to an earlier commit

1. On the main menu, choose VCS | Git | Reset HEAD. The Reset Head dialog box opens.

2. From the Git Root drop-down list, choose the required local repository.

3. Make sure the Current Branch read-only field shows the branch in which you actually want to reset the HEAD. If necessary, switch to the correct branch.

4. From the Reset Type drop-down list, select the desired reset strategy to apply:
   - To have the changed files preserved but not marked for commit, select the Mixed option. The index will be reset while the working tree will not and you will get a report of what has not been updated.

     Tip This is the default strategy.

   - To have only the HEAD pointer moved without updating the index and the working tree, select the Soft option. Your current state with any changes will remain different from the commit you are switching to.
   - To have both the working directory and the index changed to the specified commit, select the Hard option.

5. In the To Commit text box, specify the commit you want to reset the current HEAD to. Type its commit hash or use an expression, for example, of the following structure:
   `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`.

   Refer to the Git commit naming conventions for details.

6. To check that the specified commit exists and is the one you actually need, click the Validate button. The Paths affected in commit dialog box shows the files that were affected in the specified commit. View the information and click OK.
   If you specify a commit that does not exist, PhpStorm displays an error message.

Applying Changes from a Specific Commit to Other Branches (Cherry Picking)

Suppose, you have two diverged branches, `master` and `hotfix`. One day, you decide to integrate changes from one specific commit in `master` to `hotfix`. With PhpStorm, you can cherry pick between branches right from the Version Control tool window.

To cherry pick changes between two branches

1. Switch to the target branch that the changes will be integrated to.
2. Open the Version Control tool window and switch to the Log tab.
3. From the Branch drop-down list on the toolbar, select the source branch containing the changes you want to cherry-pick.
   Note that the Commits pane lists all commits performed in the selected branch. To reduce the number of items in the list, you can filter the commits by the user or date.
   You can also click the Highlight non-picked commits button to grey out the commits that have already been applied to the current branch.
4. Select the required commit. Use the information in the Commit Details area if necessary.
5. Click the Cherry-pick button on the toolbar. PhpStorm will display the Commit Changes dialog with the automatically generated commit message.
6. If you want to review the changes or even modify the code before committing it to the target branch, you can do so in the difference viewer available from this dialog.
7. When done, click Commit to cherry-pick the selected changes.

Note that if you click Cancel in the Commit Changes dialog, a separate changelist will be created with the selected changes that you can see in the Local Changes tab . You can review these changes and commit them later if necessary.

Git Branches in Multirooted Projects

On this page:

- Basics
- Asynchronous branch control
- Synchronous branch control
- Rolling back a successful operation if it fails in some of the repositories

Basics

Sometimes you may need to have several local Git repositories within a single project. Such multirooted project can be set up in the following ways:

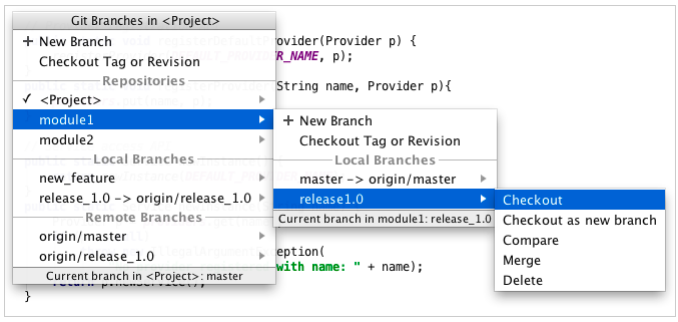
- The repositories are independent, they can be included in the project just as nested roots. One of these repositories can be intended for the project development itself while the others store the libraries used.
  Such repositories are controlled asynchronously , that is, commit, push, and other operations can be applied to them independently, just as it is done in projects with one Git repository.
- The repositories are mutually connected. Such repositories most often need to be controlled synchronously, that is, commit, push, and other operations should be applied to all these repositories at once. For such projects PhpStorm also allows you to control branches synchronously . This means that all branch operations on several repositories are performed simultaneously as if it were a single repository. In case an operation fails in at least one of the repositories, PhpStorm rolls it back in the repositories where the operation has passed successfully.

Asynchronous branch control

Asynchronous control is applied to the branches in a multirooted project when all local Git repositories in the project are independent. Commit, push, and other operations to the branches in the repositories are applied independently, just as it is done in projects with one Git repository.

To apply a Git operation to a branch asynchronously (in one of the repositories)

1. Open the Git Branches popup menu.
2. In the Git Branches popup menu that opens, choose the branch to apply the required action to. The popup menu shows a list of branches in the current repository and a list of all the repositories in the project.
   The current repository is detected in the following ways:
   - If you have explicitly selected a file or a folder in the Project tool window or elsewhere, PhpStorm treats the root under which the file is located as current repository.
   - If you used the VCS | Git | Git Branches command on the main menu, PhpStorm treats the root of the file currently opened in the editor as current repository.

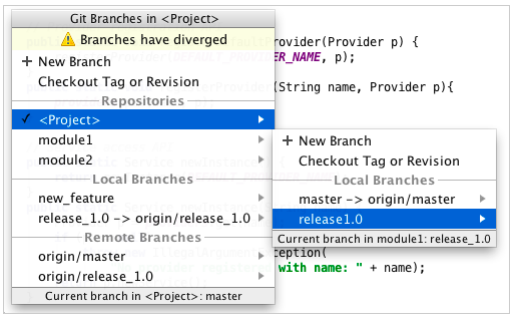


   Do one of the following:
   - To update a branch in the current repository, select the branch in the list.
   - To access another repository, select it under the Repositories group, and then select the relevant branch in the subordinate popup menu that opens.
3. Click the branch and choose the required action from the popup menu that opens.

Synchronous branch control

In a multirooted project when several mutually connected repositories, it is often helpful to apply synchronous branch control, which involves the following:

- All branch operations on several repositories are performed simultaneously as if it were a single repository. This applies to all actions including checkout , merge , and delete . If you invoke the compare action, PhpStorm shows a dialog box where you are asked to choose the repository to compare the branch with.
- The Git Branches popup lists only those local and remote branches that are common for all the repositories in the project.
- Creating a new branch from the popup will create it on all the repositories.
- You may still control branches individually for each repository by using the repository selector in the Git Branches popup menu. If you happen to checkout a branch only in one of the repositories, you'll get the warning that branches have diverged:



To disable synchronous branch control, do one of the following

When you invoke the Git Branches popup menu for the first time, PhpStorm examines the repositories registered in the project. If they all are on branches with same names, PhpStorm displays a popup menu that proposes to control branches synchronously.

- Click the Disable link in the information popup window.
- Open the Settings dialog box , and on the Git page under the Version Control node, clear the Control repositories synchronously check box.

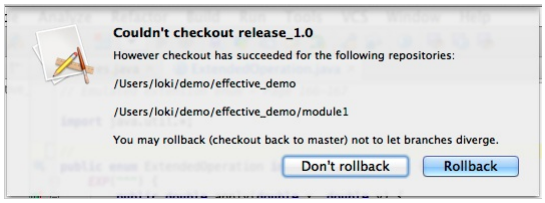
To enable synchronous branch control

- Open the Settings dialog box , and on the Git page under the Version Control node, select the Control repositories synchronously check box.

Rolling back a successful operation if it fails in some of the repositories

If you checkout a branch on several roots and the operation fails for some of the roots but has already succeeded for the others, the branches diverge. For example, the `master` branch is checked out in one repository while the `feature` branch is checked out in the other.

To prevent this, PhpStorm offers to rollback the successful checkout operation, so the previous checked out branch is checked out again.



Besides checkout, PhpStorm suggests to rollback merge and other branch operations.

Setting Up a Local Git Repository

Although Git provides high flexibility in arranging data and your work with repositories, the following scenarios are most commonly used for setting up a local Git repository:

- Clone an existing remote repository and create a new project with the downloaded data.
- Create a local repository which you can push to a remote location later, if necessary.

Warning! Git does not support external paths. So if you choose another directory, note that it must contain the tree where the project root resides.

To clone a remote Git repository

1. On the main menu, choose VCS | Checkout from Version Control | Git. The Clone Repository dialog box opens.
2. In the Git Repository URL text box, type the URL of the remote repository which you want to clone.
3. Click the Test button next to the Git Repository URL text box to check that connection to the remote repository can be established successfully.
4. In the Parent Directory text box, specify the directory where you want PhpStorm to create a folder for your local Git repository. Use ..., if necessary.
5. In the Directory Name text box, specify the name of the new folder into which the repository will be cloned. Click Clone.
6. Create a new project based on the cloned data by accepting the corresponding suggestion displayed by PhpStorm.

Git root mapping will be automatically set to the project root directory.

To create a local Git repository

1. Open the project you want to store in a repository.
2. On the main menu, choose VCS | Import into Version Control | Create Git Repository.
3. In the dialog that opens , specify the directory where you want to create a new Git repository.
4. Put the required files under Git version control. The files appear in the Local Changes tab of the Version Control tool window, under the Default changelist.

Tip By default, PhpStorm suggests the root of the current project.

- If you specify Git as the version control system for a directory in the Version Control dialog box, PhpStorm will suggest to put each new file in this directory under Git control.

Stashing and Unstashing Changes

On this page:

- Stashing and unstashing
  - Saving modifications to a new stash
  - Applying a stash
  - Removing stashes
- Stashing vs s helving

Stashing and unstashing

Sometimes it may be necessary to revert your working copy to match the HEAD commit but you do not want to lose the work you have already done. This may happen if you learn that there are upstream changes that are possibly relevant to what you are doing or if you need to make some urgent fixes.

Stashing involves recording the difference between the HEAD commit and the current state of the working directory (stash). Changes to the index can be stashed as well.

Unstashing involves applying a stored stash to a branch.

You can apply a stash to an existing branch or create a new branch on its basis.

A stash can be applied as many times as you need to any branch you need, just switch to the required branch. Keep in mind that:

- Applying a stash after a series of commits results in conflicts that need resolving.
- A stash cannot be applied to a "dirty" working copy, that is a working copy to which changes have been made since the latest commit.

You can store stashes in a list as long as you need and remove them from the list when necessary.

Saving modifications to a new stash

To save modifications to a new stash

1. On the main menu, choose VCS | Git | Stash Changes. The Stash dialog box opens.
2. Select the relevant Git root and make sure that the correct branch is checked out.
3. In the Message text box, describe the changes to be stashed.

   Tip To stash the local changes and bring the changes staged in the index to your working tree for examination and testing, select the Keep Index check box.

Applying a stash

To apply a stash to the same branch

1. On the main menu, choose VCS | Git | Unstash Changes. The Unstash Changes dialog box opens.
2. Select the Git root where you want to apply a stash and make sure that the correct branch is checked out.
3. In the Stashes list, select the relevant stash.

   Tip Examine the stash descriptions and the names of the branches where specific stashes were created to find the stash you need.

4. Click the View button to open the Paths affected in commit dialog box and find out which files are affected in the selected stash.
5. Specify additional unstash options:
   - To have the selected stash removed from the list after it is applied, select the Pop stash check box.
   - To have the stashed index modifications applied, select the Reinstate Index check box.

     Warning! This operation may fail if you have conflicts. This happens because conflicts are stored in the index, where you can no longer apply the changes as they were originally.

To create a new branch on the basis of a stash

1. On the main menu, choose VCS | Git | Unstash Changes. The Unstash Changes dialog box opens.
2. Select the Git root where you want to apply a stash.
3. In the Stashes list, select the relevant stash.
4. In the As new branch text box, type the name of the new branch to be created.

Removing stashes

To remove one or several available stashes

1. On the main menu, choose VCS | Git | Unstash Changes. The Unstash Changes dialog box opens.
2. In the Stashes list, select the stashes to be removed and click the Drop button.

   Tip To remove all stashes from the list, click the Clear button.

Stashing vs s helving

As you might have noticed, Stashing and Unstashing have much in common with Shelving and Unshelving respectively.

The only difference is in the way patches are generated and applied:

- Patches with stashed changes are generated by Git itself. To apply them later, you do not need PhpStorm.
- Patches with shelved changes are generated by PhpStorm. Normally, they are also applied through the IDE. Applying shelved changes outside PhpStorm is also possible but requires additional steps.

Pulling Changes from the Upstream (Git Pull)

Refreshing a local Git repository with the changes from the remote repository (Pull) involves retrieving changes (Fetch) and applying them to the local data (Merge). The Git integration with PhpStorm provides interface for specifying the mandatory Pull settings and for customizing the Pull procedure.

To pull changes from a remote repository

1. On the main menu, choose VCS | Git | Pull Changes. The Pull Changes dialog box opens.
2. Specify the required Git root and the URL address or alias of the source remote repository.
3. From the Branches to Merge list, select the remote branches you want to merge to the current local branch.
4. Specify the pull strategy and additional settings using the dialog box controls described in the Pull Changes dialog box reference.

Pushing Changes to the Upstream (Git Push)

The Git integration in PhpStorm supports uploading changes from the current branch to its tracked branch or to any other remote branch. If the upload (push) is rejected due to lack of synchronization between your local repository and the remote storage, you can either interactively choose the strategy to update the local branch or have PhpStorm update it silently.

Before uploading changes from your local Git repository to a remote repository make sure that:

- You have a Git hosting account.
- The target remote Git repository is accessible for your account.
- You have a pair of `ssh keys` to access the remote repository.

In this section:

- Pushing changes
- Updating a local branch interactively if push is rejected
- Configuring automatic update if push is rejected

To push changes

1. On the main menu, choose VCS | Git | Push . The Git Push dialog box opens listing all commits made in the current branch since the last `push` .
2. Select the commit to push. The right-hand pane shows the changes included in the selected commit. Use the toolbar buttons to examine the commit details.
3. Select which files you want to upload. To select all files, click the Select All toolbar button or press `Ctrl+A` .
4. Specify the target remote branch to push the changes to. If you want to upload the current branch to an alternative branch, rather than its tracked branch, click the branch name to edit it (note that completion is available for branch names).
   You can also click the Edit all targets link in the bottom right corner to edit all branch names simultaneously.
5. Click the Push button, when ready. PhpStorm starts the upload.
6. If `push` is rejected due to lack of synchronization, update the local branch in one of the following ways:
   - Choose the update strategy manually in the Push Rejected dialog box.
   - Have the local branch updated automatically .

To update a local branch interactively if push is rejected

When `push` is rejected due to lack of synchronization between the current local branch and its tracked remote branch, PhpStorm displays the Push Rejected dialog box, provided that the Auto-update if push ... rejected check box in the Git page of the Settings dialog box is cleared.

1. If your project uses several local Git repositories, specify the ones to be updated:
   - To have all the local repositories updated, no matter whether `push` has been rejected for all of them or not, select the Update non rejected repositories as well check box.
   - To have only affected repositories updated, clear the Update non rejected repositories as well check box.
2. In the Push Rejected dialog box, specify the way to update local repositories in the future:
   - To have this dialog box displayed upon every reject of `push` , clear the Remember the update method choice and silently update in future check box.
   - To have PhpStorm apply the update procedure invoked from this dialog box silently, select the Remember the update method choice and silently update in future check box.
     After you leave the dialog box, the Auto-update if push ... rejected check box in the Git page of the Settings dialog box is selected and the applied update method becomes default. To change this setting, clear the Auto-update if push ... rejected check box.
3. Invoke the update procedure rebase or merge by clicking the buttons Rebase or Merge respectively.

To have PhpStorm update a local branch automatically if push is rejected

Do one of the following:

- Open the Settings dialog box , click Git under the Version Control node, and in the in the Git page, that opens, select the Auto-update if push ... rejected check box.
  If you have not invoked any update operation from the Push Rejected dialog box before and you are enabling the check box initially, PhpStorm will synchronize the local branch silently by means of the merge operation.

  If you have previously invoked an update operation rebase or merge from the Push Rejected dialog box and selected the Remember the update method choice... check box, PhpStorm will apply your last choice.

  Accordingly, to change the "remembered" setting, clear the check box, access the Push Rejected dialog box, select the Auto-update if push ... rejected check box, whereupon invoke another update strategy.
- During an interactive update in the Push Rejected dialog box, select the Remember the update method choice and silently update in future check box, whereupon invoke the required update method `rebase` or `merge` by pressing the corresponding button. PhpStorm saves your choice and will apply it to update the conflicting local branch silently until you change the remembered setting .

Handling LF and CRLF Line Endings

On this page:

- Basics
- Enabling smart handling of LF and CRLF line separators
- Handling the problems with line separators during commit

Basics

Quite often people working in a team and using the same repository or upstream prefer different operating systems. This may result in problems with line endings, because Unix, Linux, and OS X use `LF` and Windows uses `CRLF` to denote the end of a line. PhpStorm shows the discrepancies in line endings in the Difference Viewer Dialog , so you can fix them manually.

To have Git solve such problems automatically, you need to set the `core.autocrlf` attribute to `true` on Windows and to `input` on Linux and OS X. For more details on the meaning of the `core.autocrlf` attribute, see the article Mind the End of Your Line or Dealing with Line Endings . You can change the configuration manually by running `git config --global core.autocrlf true` on Windows or `git config --global core.autocrlf input` on Linux and OS X. However, PhpStorm can analyze your configuration, warn you if you are about to commit `CRLF` into the repository, and offer to set the `core.autocrlf` setting to `true` or `input` depending on the operating system used.

Enabling smart handling of LF and CRLF line separators

To enable smart handling of LF and CRLF line separators

To enable smart handling of `LF` and `CRLF` line separators

1. Open the Settings dialog box .
2. Under the Version Control node, click Git.
3. On the Git page that opens, select the Warn if CRLF line separators are about to be committed checkbox.

Handling the problems with line separators during commit

To handle problems with the line separators during commit

1. Enable smart handling of LF and CRLF line separators . After that, PhpStorm will show the Line Separators Warning Dialog every time you attempt to commit a file with `CRLF` separators, unless you have set any related git attributes on the affected file. In the latter case, PhpStorm supposes that you clearly understand what you are doing and excludes the file from analysis.
2. When the Line Separators Warning Dialog is displayed, do one of the following:
   - To ignore the warning and commit the file with `CRLF` separators, click the Commit As Is button.
   - To have the `core.autocrlf` attribute set to `true` or `input` depending on the operating system used before commit, click the Fix and Commit button. As a result, all the `CRLF` separators will be replaced with `LF` separators and committed into the repository. Note that the reverse operation will not be performed when you download the files into your working directory, that is, no `CRLF` will appear in place of `LF` .
   - To stop the commit procedure, click Cancel.
3. To suppress showing the dialog box in the future, select the Don't warn again check box.

Using Perforce Integration

With the Perforce integration enabled, you can perform basic Perforce operations from inside PhpStorm.

Note The information provided in the topics listed below assumes that you are familiar with the basics of Perforce version control system.

In this section:

- Enabling and Configuring Perforce Integration
- Handling Modified Without Checkout Files
- Integrating Perforce Files
- Resolving Conflicts with Perforce Integration
- Showing Revision Graph and Time-Lapse View
- Using Multiple Perforce Depots with P4CONFIG
- Working Offline
- Checking Perforce Project Status
- Attaching and Detaching Perforce Jobs to Changelists

Prerequisites

- A Perforce client is installed on your computer.
- You have an account with the Perforce depot.

To start using Perforce integration, perform the following preliminary steps

1. Create a client spec using your Perforce client.
2. Get source files from the Perforce depot using your Perforce client.
3. As soon as the local working copy is on your computer, associate your local directory with Perforce .
   After that you will be able to open source files for edit, and perform the usual Perforce-related tasks using PhpStorm.

Perforce support

- When Perforce integration with PhpStorm is enabled, the Perforce item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the Perforce control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using Perforce integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated Perforce commands.
  - Error messages.

Enabling and Configuring Perforce Integration

The Perforce Integration is disabled by default. If you want to perform Perforce-related operations right from PhpStorm, enable the integration at the IDE level and associate the project root or specific directories with Perforce. The general procedure is described in the section Enabling Version Control .

In this section:

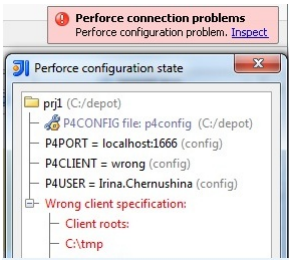
– Enabling Perforce integration for a project or directory

– Configuring Peforce integration settings

To enable Perforce integration for a project or directory

1. Open the Settings dialog box , and click Version Control.
2. In the Version Control page, that opens, point to the desired root.
   This can be <Project Root> or a content root .
3. From the VCS drop-down list, choose Perforce.

If you specify a wrong client workspace, and your project roots do not match with the workspace roots, PhpStorm displays a pop-up window with a warning.



Click Inspect to view and fix the discrepancies.

To configure Perforce integration

1. Open the Settings dialog box , and then open the Perforce page under the Version Control node.
2. To establish live connection to the Perforce server, select the Perforce is online check box.
3. Specify which credentials you want to use for connecting to the Perforce server.
   – To use the connection settings from your `P4CONFIG` files, choose the Use P4CONFIG or default connection option.
   If you are using `P4CONFIG` files for configuration, PhpStorm shows what config files it has found and what other default settings are used. This way you can be sure that your `P4CONFIG` files are detected and taken into account.
   – To configure connection manually, choose the Use connection parameters option and specify the following settings in the corresponding text boxes:
     1. The Port the Perforce client will listen to.
     2. The Client name.
     3. Your User name and Password to authenticate to the server.
4. To use ticket-based authentication, select the Login authentication check box. Otherwise, password-based authentication will be used. In either cases PhpStorm uses the login name and password specified in the dialog box or stored in the `P4CONFIG` files.
5. To attempt to log on the Perforce server without authentication, select the Try to login silently check box.
6. To have PhpStorm create a `P4.output` file and store the output of Perforce commands in it, select the Dump Perforce Commands to: check box.
7. Specify the path to the Perforce executable file. Click Test Connection to make sure your settings ensure successful connection.
8. In the Timeout field, specify the lapse of time in seconds during which PhpStorm waits for response from the server. If the server does not respond in due time, the user is prompted to disable integration.
9. To enable displaying the branch history of a specified file, including all file branch points, edits, and merges, select the Show branching history check box.
10. To have PhpStorm point at committed changes that are also integrated to other changelists and provide information on the target changelists that received the content in question, select the Show integrated changelists in committed changes check box.
11. To get the user interface for attaching and detaching Perforce jobs to changelists, select the Enable Perforce Jobs Support check box.

Handling Modified Without Checkout Files

If you are going to modify or delete a file under Perforce version control, the read-only status of such file should be removed. PhpStorm takes care of automatically making files writable. However, you can change read-only status manually, which may happen in a number of ways; for example:

- With the Clear Read-Only Status dialog enabled , you make a file writable using file system.
- When a read-only file is opened in the editor, you double-click lock icon in the status bar.
- You remove read-only attribute externally, using file properties.

In these cases, the file gets status Modified without checkout and appears in the Local Changes tab of the Version Control tool window .

To resolve 'modified without checkout' files

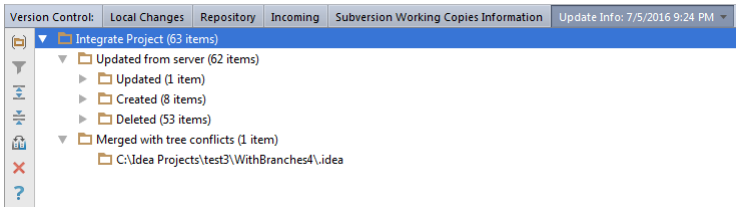
1. In the Local Changes tab of the Version Control tool window , expand Modified without Checkout node, and select the desired file.
2. On the context menu of the file, choose Check Out. The file becomes writable, and moves to the active changelist.

Integrating Perforce Files

You can merge changes between the branches into your local working copy, using the branch specification, or a changelist.

The Integrate Project command is available for both Perforce and Subversion integrations.

Integration results display in the Integrate Info tab of the Version Control tool window. Context menu of a file enables you to compare versions, view history and annotations, browse changes and more.



To integrate a Perforce branch into a project

1. On the main menu, choose VCS | Integrate Project.
2. In the Integrate dialog box, select the Perforce tab (if both Perforce and Subversion integrations are used in this project).
3. Select the sources to be merged, and the desired revision.
4. Define VCS-specific merge options.
5. Click OK.

Resolving Conflicts with Perforce Integration

As described in the section Resolving Conflicts , the conflicts might occur in course of team work. Perforce integration makes use of the following commands:

- Resolve enables you to resolve a conflict to a specific file.
- Resolve All applies to all files in a changelist that have merge status.

To resolve conflicts for the files under Perforce version control

1. In the Local Changes tab of the Version Control tool window , select a conflicting file, or a whole conflict node.
2. On the context menu of the selection, choose Resolve or Resolve All. Files Merged with Conflicts dialog box appears.
3. If you want to accept server version and overwrite your local changes, click Accept Theirs. If you want to force your changes to the repository, click Accept Yours. Clicking Merge opens the merge tool, where you can accept or discard each change individually.
4. Once the conflicts are successfully resolved, commit your local version to the repository.

Showing Revision Graph and Time-Lapse View

Using Perforce integration, you can view the Revision Graph or Time-lapse View for a file without leaving the IDE, provided that `p4v.exe` is installed on your computer.

To show Revision Graph or Time-lapse View for a file

1. Select the desired file in any navigation tool window, or open it in the editor.
2. On the main Version Control menu, or on the context menu of the selection, choose Perforce, and then select Revision Graph, or Time-Lapse View on the submenu.
3. With the first invocation, enter password in the login dialog box.

Using Multiple Perforce Depots with P4CONFIG

If your project contains directories that are stored in the different Perforce depots, you might need to switch between them. PhpStorm uses P4CONFIG to automatically switch to the respective depot as you use a Perforce-versioned directory.

P4CONFIG is an environment variable that contains the name of P4CONFIG file without a path. If a certain directory is associated with Perforce, PhpStorm seeks for P4CONFIG file in this directory and its parents; if the file is not found, it is sought in the bin directory of PhpStorm installation. When a P4CONFIG file is found, PhpStorm uses the settings contained therein, to connect to the respective Perforce depot.

A sample P4CONFG file might consist of such lines:

```
P4CLIENT=MyClient
P4USER=MySelf
P4PORT=ida:3456
```

To use multiple Perforce depots in a project, follow these general steps

1. Create a P4CONFIG file in each directory associated with Perforce.
2. Create environment variable P4CONFIG that contains file name without a path.

Working Offline

In this section:

- Offline mode basics
- Going offline
- Going online

Offline mode basics

The Perforce plugin keeps a log of VCS operations performed while offline, and replays the log when the user comes back online. The log of operations is stored in the `.iws` file and persists between PhpStorm restarts.

While offline, you can perform the following operations, which will be automatically replayed in online mode:

- Edit
- Add/Copy
- Delete
- Move/Rename
- Revert
- Move to another changelist
- View Committed/Incoming changes (displaying cached information only).

The following operations are not supported in offline mode: update, commit, integrate, tracking of the unversioned, locally deleted and *modified without checkout* files (unversioned files are shown as *unchanged*), and any other operations that require server connection.

Tip The performance of PhpStorm Perforce integration in offline mode is considerably better than in online mode (because no server calls are required), so you might want to use offline mode even though connection to the Perforce server is successful.

To go to offline mode, do one of the following

- Automatically, when the Perforce server becomes unavailable. PhpStorm switches to the offline mode automatically, and displays an offline notification in a pop-up window. To enable this behaviour, select the Switch to offline mode automatically if Perforce is unavailable check box in the Perforce page of the Settings dialog box.

- Manually at anytime, by choosing VCS | Perforce and select Work Offline on the main menu.

When offline mode is activated, the following notification balloon appears:

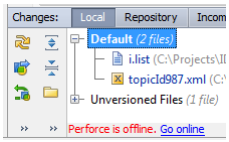
This balloon fades after a while; Perforce is offline message appears at the bottom of the Local Changes tab of the Version Control tool window.

To return to online mode, do one of the following

- Choose VCS | Perforce and clear Work Offline.

- In the offline notification balloon, click the Go online link.

- In the Local Changes tab of the Version Control tool window, click the Go online link:



Checking Perforce Project Status

Besides indicating the current file status relative to the repository, PhpStorm integration with Perforce provides you with the accumulated view of the project files' statuses.

In this section:

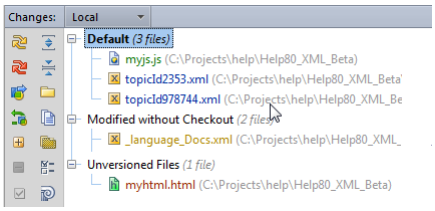
- Viewing the statuses of project files
- Refreshing file status

To view differences between the current state of the project files and the repository

1. Open the required project.

2. On the main menu, choose VCS | Refresh File Status.

3. Switch to the Version Control tool window, tab Local .



The status of each file is indicated by the color in which the path to the file is displayed.

To refresh the statuses of project files

PhpStorm provides two refresh modes for statuses of files under Perforce control.

- Standard Refresh takes into consideration only the changes made through the PhpStorm integration with Perforce. This improves performance because does not require connecting to the server. However, this approach does not let you know about the changes made outside PhpStorm, for example, right through the `p4v client` application.

- Force Refresh considers all the changes made to project, both from PhpStorm and from any other application, for example, right through `p4v client` .

1. Switch to the Version Control tool window, tab Local .

2. Do one of the following:
   - To run Standard Refresh, click the Refresh toolbar button or press `Ctrl+F5` .
   - To run Force Refresh, click the Force Refresh toolbar button .

Attaching and Detaching Perforce Jobs to Changelists

The Perforce integration with PhpStorm provides you with a user interface for attaching and detaching Perforce jobs to changelists.

Note that the integration does not support creation of Perforce jobs.

To get access to working with Perforce jobs, select the Enable Perforce Jobs Support check box on the Perforce page of the Settings dialog box.

From this topic you will learn how to:

- Attach jobs to changelists:
  - At any stage of your work from the Local tab.
  - During commit from the Commit Changes dialog box.
- Find the desired job to attach to a changelist using:
  - The standard search functionality, which enables you to specify numerous search criteria and thus to flexibly configure the search procedure.
  - The quick search functionality, which enables you to have the found job linked to the changelist automatically.

    Tip This functionality is helpful when you need to attach only one job to a changelist and you either know the exact name of the desired job or at least can specify a search pattern for the name.
- Detach jobs from changelists.

To find and link a job at any stage of your work

1. Open the Local tab of the Version Control tool window.
2. Select the changelist you want to link a job to.
3. From the context menu of the changelist, choose Edit Associated Jobs. The Edit Jobs Linked to Changelist dialog box opens.
4. Find the desired job. Do one of the following:
   - To use the standard search functionality, click the button.
   - To use the quick search functionality, click the button.
5. View the details of the found job and click OK.

To find and link a job during commit

1. In the Local Changes tab of the Version Control tool window, select the changelist you want to link a job to and open the Commit Changes dialog box.
2. Find the desired job using the controls in the Jobs area. Do one of the following:
   - To use the standard search functionality, click the button.
   - To use the quick search functionality, click the button.
3. View the details of the found job and continue the commit procedure.

To find a job using the standard search functionality

1. In the Edit Jobs Linked to Changelist dialog box or in the Jobs area of the Commit Changes dialog box, click the button.
   Which dialog box you are currently in depends on whether you are linking jobs during commit or at any other stage of work.
2. In the Link Job to Changelist dialog box that opens, specify the desired search parameters.

   Tip At least one of the fields should be filled in.
3. Click the Search button. The jobs that match the specified criteria are shown in the Search Results list. To view the details of a job, select it in the list.
4. Select the desired job and click OK. The dialog box closes and you return to the dialog box where you started the search:
   - The Commit Changes dialog box, if you are linking jobs during commit .
   - The Edit Jobs Linked to Changelist dialog box, if you are linking jobs at any other stage stage of work.

To quickly find and link one job

1. Open the Edit Jobs Linked to Changelist dialog box or switch to the Jobs area of the Commit Changes dialog box.
   The dialog box to be used depends on whether you are linking jobs during commit or at any other stage of work.
2. In the text box, type the desired job name search pattern and click the button. The job is found and automatically linked to the current changelist.
   If no job matching the specified pattern is found, the corresponding information message is displayed.

   Warning! The details of jobs that are found and linked through the quick search functionality are available only in the Edit Jobs Linked to Changelist dialog box.

To detach a job from a changelist

- In the Edit Jobs Linked to Changelist dialog box or in the Jobs area of the Commit Changes dialog box, select the desired job and click the button.
  The dialog box to be used depends on whether you are detaching jobs during commit or at any other stage of work.

Using Mercurial Integration

With the Mercurial integration enabled, you can perform basic Mercurial operations from inside PhpStorm.

> **Note** The information provided in the topics listed below assumes that you are familiar with the basics of Mercurial version control system.

In this section:

- Adding Files to a Local Mercurial Repository
- Setting up a Local Mercurial Repository
- Managing Mercurial Branches and Bookmarks
- Switching Between Working Directories
- Pulling Changes from the Upstream (Pull)
- Pushing Changes to the Upstream (Push)
- Tagging Changesets

Prerequisites

- Mercurial is installed on your computer.
- The location of the Mercurial executable file `hg.exe` is correctly specified on the Mercurial page of the Settings dialog box.
  If you followed the standard installation procedure, the default location is `/opt/local/bin` or `/usr/local/bin` for Linux and OS X and `/Program Files/TortoiseHG` for Windows.

  It is recommended that you add the path to the Mercurial executable file to the `PATH` variable. In this case, you can specify only the executable name, the full path to the executable location is not required. For more information about environment variables, see Path Variables .
- Mercurial integration is enabled for the current project root or directory.

If you want to use a remote repository, create a Mercurial hosting account first. You can access the remote repository through a pair of `ssh keys` or apply the username/password and keyboard interactive authentication methods supported by the Mercurial integration.

Mercurial support

- When Mercurial integration with PhpStorm is enabled, the Mercurial item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the Mercurial control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using Mercurial integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated Mercurial commands.
  - Error messages.

Adding Files To a Local Mercurial Repository

After a Mercurial repository for a project is initialized , you need to add the project data to it:

- If you have specified Mercurial as the version control system for your project in the Settings dialog box, PhpStorm suggests to put each new file under Mercurial control during the file creation.
  To have Mercurial ignore some types of files, configure files to ignore .
- You can add all unversioned files to Mercurial control or select files to add .

To add all currently unversioned files to Mercurial control

1. Switch to the Version Control tool window.
2. In the Local Changhes tab, navigate to the Unversioned Files node and choose Add to VCS from the context menu.

To add specific file(s) to a local Mercurial repository, do one of the following:

- Switch to the Version Control tool window, expand the Unversioned Files node, and select the files to be added. From the context menu, choose Add to VCS.
- Switch to the Project tool window and select the files to be added. From the context menu, choose Mercurial | Add to VCS.

Setting Up a Local Mercurial Repository

Although Mercurial provides high flexibility in arranging data and your work with repositories, the following scenarios are most commonly used for setting up a local Mercurial repository:

- Clone an existing remote repository and create a new project with the downloaded data.
- Create a local repository which you can push to a remote location later, if necessary.

To clone a remote Mercurial repository

1. On the main menu, choose VCS | Checkout from Version Control | Mercurial. The Clone Mercurial Repository dialog box opens.
2. In the Mercurial Repository URL text box, type the URL of the remote repository which you want to clone.
3. Click the Test Repository button next to the Mercurial Repository URL text box to check that connection to the remote repository can be established successfully.
4. In the Parent Directory text box, specify the directory where you want PhpStorm to create a folder for your local Mercurial repository. Use the Browse button button, if necessary.
5. In the Directory Name text box, specify the name of the new folder into which the repository will be cloned. Click Clone.
6. Create a new project based on the cloned data by accepting the corresponding suggestion displayed by PhpStorm.

To create a local Mercurial repository

1. Open the project you want to store in a repository.
2. On the main menu, choose VCS | Import into Version Control | Create Mercurial Repository. The Create Mercurial Repository dialog box opens.
3. Specify the location of the new repository.
   - To have the repository created in the project root, choose the Create repository for the whole project option. PhpStorm will create the `.hg` directory in the project root folder.
     This option is selected by default.
   - To have a new repository created in another location, choose the Select where to create repository option and specify the path to the repository location in the text box below. Type the path manually or click the Browse button and choose the relevant folder in the Select directory for hg init dialog box that opens.

     Warning! Mercurial does not support external paths. So if you choose another directory, note that it must contain the tree where the project root resides.

     If you choose a directory which is already under Mercurial control, PhpStorm opens the Directory Is Under hg dialog box, where you can choose to create a repository in the specified location or to stay in the parent repository.
4. Put the required files under Mercurial version control. The files appear in the Version Control tool window under the Default node.
   Note that if you specify Mercurial as the version control system for a directory in the Version Control dialog box, PhpStorm will suggest to put each new file in this directory under Mercurial control.

Managing Mercurial Branches and Bookmarks

With PhpStorm, you can use both named branches and light-weight branches (bookmarks) . PhpStorm provides interface for creating, merging, and switching between branches and bookmarks, see Switching Between Working Directories . You can also run commands in the embedded Terminal, see Working with Embedded Local Terminal .

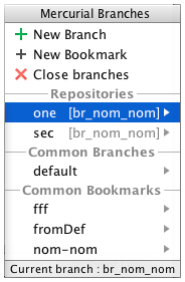
On this page:

- Opening the Branches pop-up list
- Creating a named branch
- Creating a bookmark
- Closing a branch
- Merging named branches and bookmarks
- Merging a named branch or bookmark with another named branch or bookmark
- Merging a named branch or bookmark with a changeset

For information about switching between branches and bookmarks, see Switching Between Working Directories .

Opening the Branches pop-up list

Most of the operations with branches and bookmarks are invoked from the Branches pop-up list.



The list shows all the Mercurial repositories under the project root, all the named branches, and all the bookmarks in them. The current repository and the current bookmark are marked with a tick. The name of the current named branch is shown in the dedicated hg area on the Status bar:

To open the Branches pop-up list, do one of the following:

- On the Status bar, click the name of the current named branch in the dedicated hg area.

- On the main menu, choose VCS | Mercurial | Branches.
- On the context menu of the Editor or Version Control tool window, choose Mercurial | Branches.

Creating a named branch

1. In the Branches pop-up list, click New Branch.
2. In the Create New Branch dialog box that opens, specify the name of the new branch.

The new branch immediately becomes active and its name is shown on the Status bar in the hg area.

Creating a bookmark

1. In the Branches pop-up list, click New Bookmark.
2. In the New Bookmark dialog box that opens, specify the name of the bookmark to be created.
3. Specify, whether you want to switch to the new bookmark immediately or not.
   - To activate the new bookmark and thus enable tracking and updating the light-weight branch the bookmarks identifies, leave the Inactive check box cleared. The new bookmark immediately becomes active and its name is marked with a tick in the Branches pop-up list.
   - To have an inactive bookmark created, that is, to remain in the current light-weight branch (bookmark) or named branch and switch to the new bookmark later, select the Inactive check box.

Closing a branch

According to Mercurial workflows , when you are done with a feature development and do not expect any further changes, you close the corresponding branch. A closed branch is not displayed among active branches, in the Log view , etc. To close a branch, do the following:

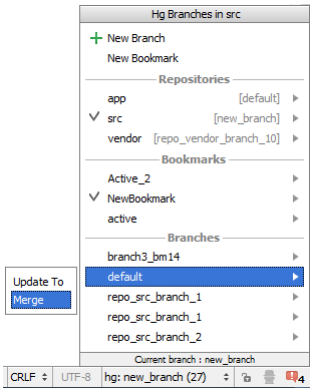
1. In the Branches popup, click Close branch. The Commit changes dialog will be displayed.
2. Click Commit and Close. All changes will be committed and the current branch will be closed.

Note that if you have several repositories listed in the Repositories section, the corresponding menu option will toggle to Close branches and the `close` operation will be applied to all of them.

Merging named branches and bookmarks

You can merge a named branch or a bookmark with another named branch, another bookmark, or a specific changeset identified by a tag or a revision number.

- Merging a named branch or a bookmark with another named branch or bookmark means merging with its head.
  Merging with named branches and bookmarks can be invoked through the menu item VCS | Mercurial | Merge, which opens the Mercurial-specific Merge dialog box of from the Branches pop-up list.



- Merging a named branch or a bookmark with a changeset means merging the branch head with the specified changeset. A changeset can be identified either by a revision number or a tag, see http://mercurial.selenic.com/wiki/Tag?highlight=%28\bCategoryGlossary\b%29 .
  Merging a named branch or a bookmark with a specific changeset can be invoked only through VCS | Mercurial | Merge.

For definitions and Mercurial-specific details regarding the merge operation itself, see http://www.selenic.com/hg/help/merge .

By default, Mercurial requires that before merge the current working directory should be clean, that is, it should not contain any uncommitted changes. Otherwise the merge operation fails and PhpStorm shows the corresponding error message. The message also recommends that you clean the current working directory by running the `hg merge <target branch, bookmark, or changeset> -C` to discard the uncommitted changes.

If your current working copy is not clean, you can either commit the changes or shelve them as described in Shelving and Unshelving Changes .

Merging a named branch or bookmark with another named branch or bookmark

Merging a named branch or a bookmark with another named branch or bookmark means merging with its head.

1. Make sure, your current working directory is clean, that is, it does not contain any uncommitted changes. Commit or shelve the changes, if any.

2. Invoke merge by doing one of the following:
   - In the Branches pop-up list, click the name of the branch or bookmark to merge with, then choose Merge on the pop-up menu:
   - Choose VCS | Mercurial | Merge on the main menu or Mercurial | Merge on the context menu of the Editor.
     In the Merge dialog box that opens:
     1. Choose the target repository from the Repository drop-down list which shows all the Mercurial repositories available under the current project root.
     2. Choose the Branch or Bookmark option and choose the named branch or bookmark to merge the current working directory with.

3. Resolve conflicts. As soon as a conflict takes place, the Files Merged with Conflicts dialog box opens with a list of conflicting files. Use the controls of the dialog box to resolve the problems:
   - To have the version of the current working directory preserved, click Accept Yours.
   - To have the version of the branch you are merging with preserved, click Accept Theirs.
   - To resolve the conflicts manually, click Merge and use the Conflict Resolution Tool, as described in Resolving Conflicts .

   If no conflicts arise during merge, the operation passes silently and the merge log is shown in the Version Control tool window.

Merging a named branch or bookmark with a changeset

Merging a named branch or a bookmark with a changeset means merging the branch head with the specified changeset. A changeset can be identified either by a revision number or a tag, see http://mercurial.selenic.com/wiki/Tag?highlight=%28\bCategoryGlossary\b%29 .

1. Make sure, your current working directory is clean, that is, it does not contain any uncommitted changes. Commit or shelve the changes, if any.

2. Choose VCS | Mercurial | Merge on the main menu or Mercurial | Merge on the context menu of the Editor.

3. In the Merge dialog box that opens:
   1. Choose the target repository from the Repository drop-down list which shows all the Mercurial repositories available under the current project root.
   2. Choose the Tag or Revision option and choose the tag or specify the hash or revision number to merge the current working directory with. To copy a hash, open the Log tab of the Version Control tool window, select the relevant branch and revision, and then choose Copy Hash on the context menu of the selection.

4. Resolve conflicts. As soon as a conflict takes place, the Files Merged with Conflicts dialog box opens with a list of conflicting files. Use the controls of the dialog box to resolve the problems:
   - To have the version of the current working directory preserved, click Accept Yours.
   - To have the version of the branch you are merging with preserved, click Accept Theirs.
   - To resolve the conflicts manually, click Merge and use the Conflict Resolution Tool, as described in Resolving Conflicts .

   If no conflicts arise during merge, the operation passes silently and the merge log is shown in the Version Control tool window.

Switching Between Working Directories

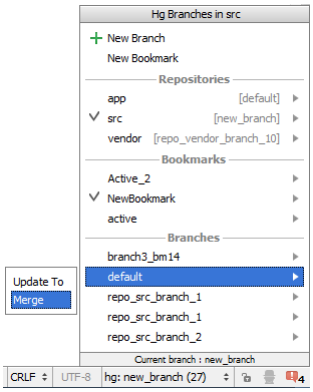
The Mercurial integration with PhpStorm provides the possibility to switch update the repository's working directory to the specified changeset or a specific line of development . Changesets can be identified by their hashes or by previously assigned tag identifiers .

On this page:

- Opening the Branches pop-up list
- Switching to another named branch or bookmark
- Switching to another changeset

You can update a named branch or a bookmark to another named branch, another bookmark, or a specific changeset identified by a tag or a revision number.

- Updating a named branch or a bookmark to another named branch or bookmark means updating to its head.
  Updating to named branches and bookmarks can be invoked through the menu item VCS | Mercurial | Update to, which opens the Mercurial-specific Switch Working Directory dialog box of from the Branches pop-up list.



- Updating a named branch or a bookmark to a changeset means updating the branch head to the specified changeset. A changeset can be identified either by a revision number or a tag, see http://mercurial.selenic.com/wiki/Tag?highlight=%28\bCategoryGlossary\b%29 .
  Updating a named branch or a bookmark to a specific changeset can be invoked only through VCS | Mercurial | Update to.

By default, Mercurial requires that before update the current working directory should be clean, that is, it should not contain any uncommitted changes. Otherwise the update operation fails and PhpStorm shows the corresponding error message. The message also recommends that you clean the current working directory by running the `hg update <target branch, bookmark, or changeset> -C` to discard the uncommitted changes.

If your current working copy is not clean, you can either commit the changes or shelve them as described in Shelving and Unshelving Changes . PhpStorm provides the possibility to discard any uncommitted changes when the update operation is already invoked. This option is available only in the Mercurial-specific Switch Working Directory dialog box.

Opening the Branches pop-up list

To open the Branches pop-up list, do one of the following:

- On the Status bar, click the name of the current named branch in the dedicated hg area.

- On the main menu, choose VCS | Mercurial | Branches.
- On the context menu of the Editor or Version Control tool window, choose Mercurial | Branches.

Switching to another named branch or bookmark

Updating a named branch or a bookmark to another named branch or bookmark means updating to its head.

1. Make sure, your current working directory is clean, that is, it does not contain any uncommitted changes. Commit or shelve the changes, if any.
   If you invoke update through the Switch Working Directory dialog box, you can also prevent conflicts by having any uncommitted changes discarded.
2. Invoke update by doing one of the following:
   - In the Branches pop-up list, click the name of the branch or bookmark to update to, then choose Update to on the pop-up menu:
   - Choose VCS | Mercurial | Update to on the main menu or Mercurial | Update to on the context menu of the Editor.
     In the Switch Working Directory dialog box that opens:
     1. Choose the target repository from the Repository drop-down list which shows all the Mercurial repositories available under the current project root.
     2. Choose the Branch or Bookmark option and choose the named branch or bookmark to update the current working directory to.
     3. To prevent failures during update if the current working directory is not clean, select the Overwrite locally modified files (no backup) check box. The uncommitted changes will be discarded.

Switching to another changeset

1. Make sure, your current working directory is clean, that is, it does not contain any uncommitted changes. Commit or shelve the changes, if any.
   If you invoke update through the Switch Working Directory dialog box, you can also prevent conflicts by having any uncommitted changes discarded.
2. Invoke update by doing one of the following:
   - In the Branches pop-up list, click the name of the branch or bookmark to update to, then choose Update to on the pop-up menu:
   - Choose VCS | Mercurial | Update to on the main menu or Mercurial | Update to on the context menu of the Editor.
     In the Switch Working Directory dialog box that opens:
     Choose the target repository from the Repository drop-down list which shows all the Mercurial repositories available under the current project root.

1. Choose the target repository from the Repository drop-down list which shows all the Mercurial repositories available under the current project root.
2. Choose the Branch or Bookmark option and choose the named branch or bookmark to update the current working directory to.
3. To prevent failures during update if the current working directory is not clean, select the Overwrite locally modified files (no backup) check box. The uncommitted changes will be discarded.

3. Resolve conflicts. As soon as a conflict takes place, the Files Merged with Conflicts dialog box opens with a list of conflicting files. Use the controls of the dialog box to resolve the problems:
   - To have the version of the current working directory preserved, click Accept Yours.
   - To have the version of the branch you are merging with preserved, click Accept Theirs.
   - To resolve the conflicts manually, click Merge and use the Conflict Resolution Tool, as described in Resolving Conflicts .

   If no conflicts arise during update, the operation passes silently and the update log is shown in the Version Control tool window.

Pulling Changes from the Upstream (Pull)

Refreshing a local Mercurial repository with the changes from the remote repository (Pull) involves retrieving changes and applying them to the local data . The Mercurial integration with PhpStorm provides interface for specifying the mandatory Pull settings and for customizing the Pull procedure.

To pull changes from a remote repository

1. On the main menu, choose VCS | Mercurial | Pull Changesets. The Pull dialog box opens.

2. Specify the required URL address of the source remote repository.

Pushing Changes to the Upstream (Push)

Before uploading (Pushing) changes from your local Mercurial repository to a remote repository make sure that:

- You have a Mercurial hosting account.
- The target remote Mercurial repository is accessible for your account.
- You have a pair of `ssh keys` to access the remote repository.

To push changes from a local Mercurial repository

1. On the main menu, choose VCS | Mercurial | Push Changeset. The Push dialog box opens.
2. In the Destination Repository URL text box, specify the location of the remote repository to push.
3. Specify the changes to upload:
   - To have a branch uploaded entirely, select the Branch check box and choose the relevant branch from the drop-down list.
   - To have a specific changeset uploaded, select the Revision check box and specify the desired revision number in the text box. By default, the latest revision (tip) is uploaded. Learn more about changesets and tips at http://mercurial.selenic.com/wiki/UnderstandingMercurial .
4. To push a locally created branch or to avoid updating before upload, select the Force Push check box.

Tagging Changesets

PhpStorm supports both local and global tags. Local tags are stored in the file `.hg/localtags` in the repository, global tags are stored in the file `.hgtags`.

Currently tagging specific changesets is supported only in the command line mode in the embedded Terminal. To launch the Terminal, hover your mouse pointer over in the lower left corner of the IDE, then choose Terminal from the menu (see Working with Embedded Local Terminal for details). After commit, tags appear in the Log tab of the Version Control tool window.

For more information about Mercurial tags, see Mercurial Documentation: Tag .

Tagging a repository

PhpStorm provides UI for tagging the current repository which means assigning a tag to its tip. The created tag is global, is stored in the file `.hgtags`. After commit, the tag appears in the Log tab of the Version Control tool window.

1. Open the Tag dialog box by doing one of the following:
   - On the main menu, choose VCS | Mercurial | Tag Repository.
   - On the context menu of the Editor, choose Mercurial | Tag Repository.

2. In the Tag dialog box that opens, specify the tag name. The name must be unique.

Using Subversion Integration

With the Subversion integration enabled, you can perform basic Subversion operations from inside PhpStorm.

PhpStorm currently supports integration with Subversion 1.9 and below.

PhpStorm comes bundled with the Subversion plugin. If you are using SVN 1.7 or below, this plugin is enough for Subversion integration. If you are using SVN 1.8 or higher, you also need to download and install the command line client on your machine. In this case, make sure the Use command line client option is selected in the Subversion settings page .

Subversion support

- When Subversion integration with PhpStorm is enabled, the Subversion item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the Subversion control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using Subversion integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated Subversion commands.
  - Error messages.

Authenticating to Subversion

The Subversion server does not require user authentication on every request. When you use Subversion integration in PhpStorm, you only need to answer the authentication challenge of the server if it is required by the authentication and authorization policies. Upon successful authentication, your credentials are saved on disk, in `~/.subversion/auth/` on Unix systems or `<USER HOME>/.subversion_IDEA` on Windows and OS X.

When an authentication challenge comes from the server, the credentials are sought for in the disk cache; if the appropriate credentials are not found, or fail to authenticate, you are prompted to specify your login and password.

If necessary, you can opt to delete all credentials stored in the cache for the http, svn and ssh+svn protocols.

To delete credentials from disk

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Version Control.

2. Open the Subversion settings page and click the Clear Auth Cache button.

Browsing Subversion Repository

Prior to checking files out, you can browse the contents of a Subversion repository. The Subversion Repository browser enables you to add or discard repository locations, view the history of files and folders, check out files and folders, navigate to the source code, browse changes, create branches or tags, etc.

Browsing the contents of a Subversion repository is always available, even when Subversion is not enabled in your project. All you need is a valid user account.

To browse the contents of a Subversion repository

1. On the main menu, choose VCS | Browse VCS Repository | Browse Subversion Repository.
2. The SVN Repositories tool window will open.
3. If no repositories have been specified so far, click + in the toolbar and specify the repository URL in the New Repository Location dialog that opens.
4. Browse the repository. If authentication is required, enter your credentials in the Authentication Required dialog that opens when you first try to expand the top node.

Checking Out Files from Subversion Repository

By checking out files from a Subversion repository, you obtain a local working copy of the repository , which you can edit. After making the necessary changes, you can publish the results by committing, or checking in your changes to the repository.

To check out files from a Subversion repository, do the following:

1. On the main menu, choose VCS | Checkout from Version Control | Subversion.
2. In the Check Out From Subversion dialog box, expand the desired repository location and select the element you want to check out.
3. Click the Checkout button.
4. In the dialog that opens , specify the destination directory where the local copy of the repository files will be created, and click OK.
   If you are checking out sources for an existing project, the destination folder should be below the project content root .
5. In the SVN Checkout Options dialog box, specify the following settings:
   - Revision to be checked out (HEAD or a selected revision).
   - Whether you need to check out the nested directories.
   - Whether you need to include the external locations.

   Click OK.

Tip This action is also available from the SVN Repositories tool window . Right-click a directory and choose the required command from the context menu.

PhpStorm suggests to create a project based on the sources checked out from version control.

If you accept the suggestion, open new project, as described in the section Opening Multiple Projects .

Cleaning Up Local Working Copy

The Cleanup command in Subversion can be helpful in the following situations:

- Your local working copy is in an inconsistent state because a Subversion command was interrupted.
- The timestamp of a file has changed while its content remains intact.

To clean up the local working copy, do one of the following:

- Select the desired file or directory in the Project tool window and choose Subversion | Cleanup from the context menu of the selection.
- Open the desired file in the editor and choose VCS | Subversion | Cleanup from the main menu.
- Select the desired file or directory in the Local Changes tab of the Version Control tool window and choose Subversion | Cleanup from the context menu of the selection.

Comparing With Branch

In addition to the common file versions comparison options, the Subversion integration with PhpStorm provides a special command that enables you to compare a file from your local working copy with its version in the selected branch.

To compare a file with its version in a specified branch, do the following:

1. Select the desired file in the Project tool window , or open it in the editor.
2. From the main VCS menu, or on the context menu of the selection, choose Subversion | Compare with Branch.
3. In the Compare with Branch pop-up, select the desired branch. The Compare with Branch dialog in the form of the Differences viewer for files appears.

Tip This action is also available in the SVN Repositories browser. Right-click the desired directory and choose the corresponding command from the context menu.

Configuring the Format of the Local Working Copy

A working copy is a directory that contains a collection of files which you can use as your private work area, as well as some extra files, created and maintained by Subversion. For example, each directory in your working copy contains an administrative directory named `.svn`.

You can have local working copies created with Subversion 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, or 1.9. PhpStorm handles all these formats, giving you a choice to upgrade to the new format or preserve the legacy one.

Switching to another Subversion format is always associated with a specific working copy (directory) under the Subversion control. In other words, one cannot upgrade to a newer Subversion format at the IDE level, but only to a directory under Subversion control.

> **Tip** To check VCS association, go to File | Settings | Version Control where all mappings between the project directories (or the entire project) and VCSs are listed.

To change the format of the local working copy, do the following:

1. Open the Version Control tool window by doing one of the following:
   - On the main menu, choose View | Tool Windows | Version Control.
   - Press `Alt+9`.
2. Switch to the Subversion Working Copies Information tab.
   This tab is only available, when the current project sources are entirely or partially under Subversion control.
3. Scroll to the information on the required directory and click the Change link.

   > **Note** Note that Subversion 1.9 can be used with the local working copy version 1.8, so in this case the Change link will not appear.
4. In the Convert Working Copy Format dialog box, that opens, select the desired format option.

Configuring HTTP Proxy

Subversion stores the http proxy settings in the `servers` file in the user's runtime configuration area (`~/.subversion/auth/` on Unix systems or `<USER HOME>/Application Data/Subversion/auth/` on Windows).

You can configure the Subversion proxy settings in two ways:

- Edit the servers file manually.
- Configure proxy settings directly from PhpStorm. Do the following:
  1. In the Settings dialog, go to Version Control | Subversion and open the Network tab.
  2. Click the Edit Network Options button and specify the proxy settings in the Edit Subversion Options Related to Network Layers dialog box that opens.

Configuring Subversion Repository Location

Subversion repository locations are global PhpStorm properties. It means that the configured repository locations will be available no matter if a project is open or not, which is useful if you need to check out an entire project from Subversion. You can define multiple Subversion repository locations for future use.

To configure a Subversion repository location, do the following:

1. Open the SVN Repositories tool window by choosing VCS | Browse VCS Repository | Browse Subversion Repository from the main menu.
2. In the SVN Repositories tool window choose New | Repository Location from the context menu, or click the + button on the toolbar.
3. In the New Repository Location dialog, specify the repository URL.

Configuring Subversion Branches

PhpStorm allows you to compose a list of parent folders of the branches you work with. This list will be displayed every time you perform any operation with branches, for example, when you synchronize your local working copy, compare branches, etc.

Branches are configured in the Configure Subversion Branches dialog box.

To configure Subversion branches, do the following:

1. Acc the Configure Subversion Branches dialog : in the Version Control tool window, switch to the Subversion Working Copies Information tab, and then click the Configure Branches link.
2. In the Trunk location field, specify the URL of your repository trunk. Type the address or click the Browse button ... and select the trunk location in the Select Repository Location dialog that opens. This dialog shows a tree of all branches and tags under the repository root .
3. In the Branch locations area, make up a list of folders where the branches you need in your work are stored. Use the + and — buttons to add/remove branches to/from the list.

Creating Branches and Tags

PhpStorm allows you to create branches or tags on the basis of your local working copies, or their repository versions.

To create a branch or a tag in a Subversion repository, do the following:

1. From the main menu, choose VCS | Subversion | Branch or Tag. Alternatively, select the source folder in the SVN Repositories tool window and choose the Branch or Tag command from the context menu.
2. In the Create Branch or Tag dialog that opens, in the Copy From section, specify the source folder that will be copied to a branch or a tag. You can use your local working copy, or a repository location.

   If a repository location is selected as the source:

   - Click to fill the Repository Location field with the path to the project location.
   - Specify the revision to base the new branch on. This can be the HEAD revision, or a revision with the specified number. If the Specified option is selected, type the revision number, or click ... and find the desired revision in the Subversion Changes Browser.
3. In the Copy To section, specify the destination where the branch or the tag will be created. If you use the base URL, specify the name of the new branch or tag. If you opt to create a branch or tag in another repository location, type its URL, or click ... and select the destination from the Select Repository Location dialog.
4. Optionally, enter a comment a click OK.

Exporting Information From Subversion Repository

You might need to obtain a *clean* local copy of the Subversion working tree without the `.svn` catalogs. Instead of checking files out and then manually deleting the administrative directories, you can use the Export command available in the Subversion repository browser.

To export a directory from a Subversion repository, do the following:

1. In the main menu, select VCS | Browse VCS Repository | Browse Subversion Repository to open the the SVN Repositories tool window.
2. Right-click a directory you want to export and choose Export from the context menu.
3. In the Select Path dialog that opens, specify the destination directory and click OK.
4. In the SVN Export Options dialog that opens, check the Export and Destination paths and specfiy the following options:
   - Depth: use this drop-down list to specify the range of recursion into Subversion subdirectories. The available options are:
     - working copy: select this option to get files/directories from the repository subtrees that have not been checked out yet.
     - empty: select this option to involve only the current file.
     - files: select this option to involve files from the current folder.
     - immediates: select this option to involve direct children of the current file.
     - infinity: select this option to enable full recursion.
   - Replace existing files: select this option to replace files in the destination directory with the exported sources.
   - Include external locations: select this option to include external references into the export.
   - Override 'native' EOLs with:: use this drop-down list if you want to override the `svn:eol-style=native` property. This is useful if team members sharing the same repository use different operating systems, which may result in problems with line endings. The following options for line separators are available:
     - None: this option is selected by default, and keeps the `svn:eol-style=native` property unchanged.
     - LF: select this option if you are using unix
     - CRLF: select this option if you are using Windows
     - CR: select this option if you are using OS X

Importing a Local Directory to Subversion Repository

You can import an entire directory to your Subversion repository provided that you have access rights. This is helpful for putting a whole project under version control.

Import to the repository is always available, even if Subversion integration is not enabled for your project.

To import a directory to a Subversion repository, do the following:

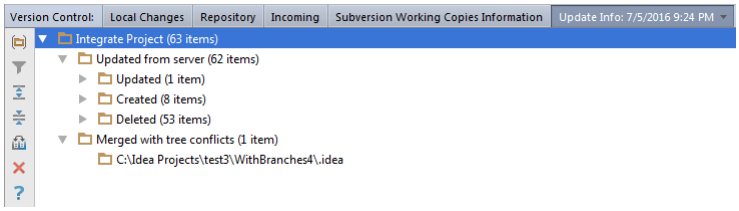
1. From the main menu, choose VCS | Import into Version Control | Import into Subversion.
2. In the Import Into Subversion dialog that opens, select the target Subversion repository. If the desired target repository location is not in the list, click the + button to add it (for more details, see Configuring Subversion Repository Location). Click Import.
3. In the Select Path dialog that opens, specify the directory you want to import and click OK.
4. In the SVN Import Options dialog that opens, check the Import to and Import from paths and specfiy the following options:
   - Depth: use this drop-down list to specify the range of recursion into Subversion subdirectories. The available options are:
     - working copy: select this option to get files/directories from the repository subtrees that have not been checked out yet.
     - empty: select this option to involve only the current file.
     - files: select this option to involve files from the current folder.
     - immediates: select this option to involve direct children of the current file.
     - infinity: select this option to enable full recursion.
   - Include ignored resources: select this option to include files that have been added to the Subversion ignore list and are not subject to version control into the import.
5. Optionally, enter a commit message, or select one from the Recent Messages drop-down list and click OK.

Integrating SVN Projects or Directories

Integrating projects or directories in Subversion means merging the differences between the two specified revisions into your working copy.

The Integrate Project command is available for both Subversion and Perforce.

Integration results are displayed in the Update Info tab of the Version Control tool window . The context menu of a file allows you to compare versions, view file history and annotations, browse changes, etc.



To integrate different sources into one Subversion project, do the following:

1. From the main menu, choose VCS | Integrate Project. The Integrate Project dialog opens.
2. If both Subversion and Perforce are used as version control systems in your project, select the Subversion tab.
3. In the Source 1 and Source 2 fields, specify the sources to be merged and select the revision. If you check the Specified option, you can click the Browse button and select a revision from the Changes Browser.
4. If necessary, select the following merge options and click OK:
   - Use ancestry: if this option is selected, ancestry of files will be noticed (this corresponds to the `svn merge` command). If unchecked, any relations between files and directories will be ignored (corresponds to `svn diff`).
   - Try merge but make no changes: select this option to preview merge results by enabling the `--dry--run` option of the SVN command. If unchecked, sources will be merged silently.
   - Depth: use this drop-down list to specify the range of recursion into Subversion subdirectories. The available options are:
     - working copy: select this option to get files/directories from the repository subtrees that have not been checked out yet.
     - empty: select this option to involve only the current file.
     - files: select this option to involve files from the current folder.
     - immediates: select this option to involve direct children of the current file.
     - infinity: select this option to enable full recursion.

Locking and Unlocking Files and Folders

Though Subversion integration allows you to successfully modify and merge files changed by different team members, sometimes it makes sense to lock files (for example, images) to avoid overwriting changes.

To lock a file:

1. Select the file you want to lock in the Project tool window or open it in the editor.
2. From the main menu, select VCS | Subversion | Lock from the main menu, or Subversion | Lock from the context menu of the selection.
3. Enter a lock comment in the Lock File dialog that opens.

To unlock a file, select the file you want to unlock, or open it in the editor, and choose VCS | Subversion | Unlock from the main menu, or Subversion | Unlock from the context menu of the selection.

Tip To forcibly break a lock set by somebody else, select the Steal Existing Lock check box.

Resolving Text Conflicts

If a conflict occurs in a file under the Subversion version control, conflict markers are added to the conflicting file, and three auxiliary unversioned files are created in your local working copy:

- filename.mine: the copy of your local file without conflict markers.
- filename.rOld: the base revision you have last synchronized to.
- filename.rNew: the latest version on the server.

Conflicting files are marked with red in the Local Changes tab of the Version Control tool window. In the Update Info tab, they are grouped in the Merged with conflicts list and are also marked with red.

With PhpStorm, you can resolve conflicts in two ways:

- Semi-automatically, using a merge tool.
- Manually in the editor. After that, you need to manually mark the processed files as conflicts-free.

On this page:

- To resolve a text conflict using the merge tool
- To resolve a text conflict manually
- To mark a file as resolved

To resolve a text conflict using the merge tool

1. In the Local Changes tab of the Version Control tool window, select the conflicting file:

2. On the main VCS menu, or on the context menu of the selection, choose Subversion | Resolve Text Conflict. The Files Merged with Conflicts dialog appears.
3. If you want to accept the server version and overwrite your local changes, click Accept Theirs. If you want to force your changes to the repository, click Accept Yours. Clicking Merge opens the merge tool, where you can accept or discard each change individually. As a result, the file is automatically marked as resolved, and auxiliary files are deleted.
4. Once the conflicts have been successfully resolved, commit your local version to the repository.

To resolve a text conflict manually

Open the conflicting file in the editor, and do one of the following:

- Edit the contents within the conflict markers as required.
- Copy one of the auxiliary files on top of your working file.

To mark a file as resolved

1. Do one of the following:
   - Select the file in the Project tree or in the Local Changes tab of the Version Control tool window, and choose Subversion, and then choose Mark Resolved on the context menu of the selection.
   - With the conflicting file opened in the editor, right-click the mouse anywhere in the editor tab. On the context menu choose Subversion, and then choose Mark Resolved.
   - On the main menu, choose VCS | Subversion | Mark Resolved..
2. In the Mark Resolved dialog box that opens select the file in question.
3. Click the Mark Resolved button.

Sharing Directory

PhpStorm supports the Subversion sharing directories feature. The Share Directory command is applied to the content roots. When such content root is shared, a new directory is created in the specified location of the Subversion repository, checked out to the local directory you want to share, and the contents of the local directory is added to the repository. As a result, the local directory contains a valid working copy of the repository.

To share a directory via the Subversion repository, do the following:

1. In the Project tool window , select an unversioned directory that is the content root of a module associated with Subversion.
2. From the main menu, select Subversion | Share Directory.
3. Specify the target repository location where the directory shall be shared.

Working With Subversion Properties for Files and Directories

Subversion integration enables you to work with Subversion-specific properties without leaving PhpStorm.

Once defined, the properties of a file or a directory are displayed in the SVN Properties view. In this view you can explore and change the existing properties and their values, or create new ones using the toolbar buttons or context menu commands.

This section describes how to:

- View properties of a file or directory from within PhpStorm
- Create a new property, or change the value of an existing property
- Set up a keyword property
- Delete a property
- Resolve property conflicts

To view the properties of a file or directory

1. In the Project tool window , select the desired file or directory under the SVN version control.
2. From the main VCS menu, or from the context menu of the selection, choose Subversion | Edit Properties. The SVN Properties view will open showing the properties of the selected file:
3. Use the toolbar buttons or the context menu commands to create, edit or delete properties, as described in the procedures below.

> **Tip** If you want the SVN Properties view to preserve its contents as you navigate through your project or edit files, make sure that the Follow Selection button is not pressed; otherwise the view will show the properties for the currently selected or edited file.

To create a new property, or set the value for an existing property

1. Open the SVN Properties view .

   > **Tip** Use the Set property command on the Subversion menu to define a single property.

2. Click the add button + on the toolbar of the SVN Properties view, or choose the Add property command on the context menu. The Set Property dialog box appears.
3. In the Property name field, type the name of the new property, or select one from the drop-down list.
4. Choose the Set property value option, and specify the desired value in the text area below.
5. To apply the changes to all subdirectories of the selected directory, select the Update properties recursively check box.
6. Click OK.

To set up the svn:keywords property

- In the SVN Properties view for a file, click .
- In the SVN Keywords dialog box, check the keywords to be included in the property.
- Click OK.

To delete a property

1. Select the property you want to delete.
2. Click – on the toolbar.

You can also delete a property from the Set Property dialog. To do this:

1. In the Property name field, select the property to be deleted.
2. Select the Delete property radio-button.
3. If you want this property to be deleted from all files and subdirectories of the selected directory, select the Update properties recursively option.

Viewing Differences in Properties

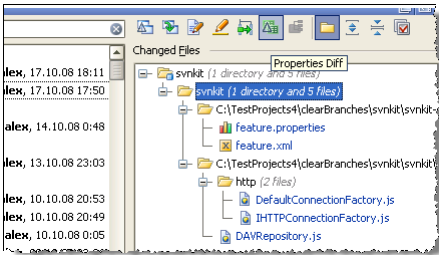
With PhpStorm, you can view differences in properties between a file in the local copy and in the repository or between two revisions of a file in the local copy.

To view property difference between the local copy and the repository version

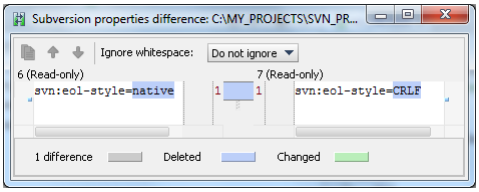
1. Open the Version Control tool window and switch to the Repository tab.

2. In the Changed Files pane, select the file for which you want to view property differences.

3. Choose Properties Diff on the context menu of the selection or click on the toolbar.



4. In the Subversion properties difference viewer, explore the differences:

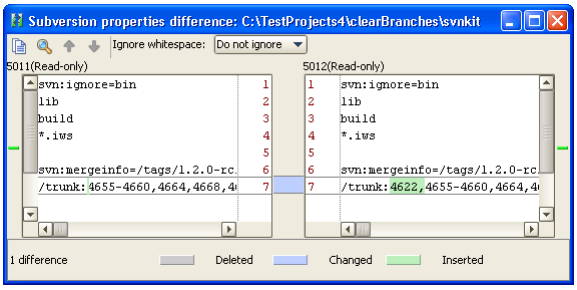


To view property difference between two revisions in the local copy

1. Open the Version Control tool window and switch to the Local tab.

2. Select the file for which you want to view property differences between revisions and choose Subversion | Properties Diff with Local in the context menu.

3. In the Subversion properties difference viewer explore the differences:



Resolving Property Conflicts

A property conflict is reported during synchronization with the server when PhpStorm detects differences between the properties of a local file or folder and their version on the server. PhpStorm does not attempt to resolve property conflicts automatically and displays the files and folders with such conflicts under the Merged with property conflicts node in the Update Info tab of the Version Control tool window. You have to resolve property conflicts manually and then tell PhpStorm to treat the corresponding files and folders as conflict-free.

On this page:

– Resolving a property conflict

– Marking a file as resolved

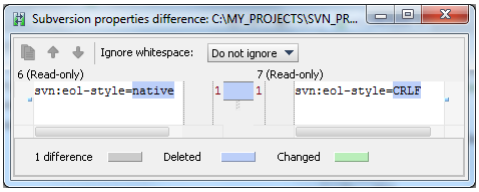
To resolve a property conflict

1. Open the Version Control tool window and switch to the Repository tab.

2. In the Changed Files pane, select the conflicting file.

3. Choose Properties Diff on the context menu of the selection or click on the toolbar.

4. In the Subversion properties difference viewer, explore the differences:



5. Based on the detected discrepancies, update the properties so the conflict is resolved.

To mark a file as resolved

1. In the Update Info tab of the Version Control tool window, select the fixed file under the Merged with property conflicts node. As you can see, the file is still displayed in red as conflicting.

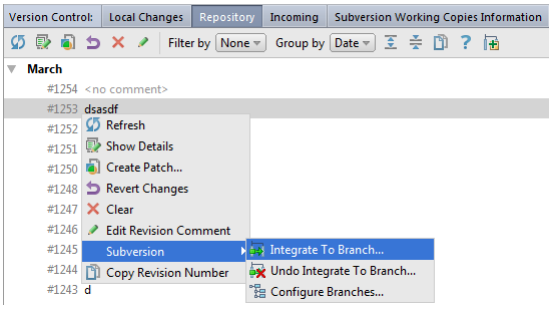
2. On the context menu of the selection, choose Subversion, and then choose Mark Resolved. When the dialog box is closed, the Local Changes tab of the Version Control tool window shows the affected files as updated. and available for checking in to the server.

3. Check in the resolved files.

Integrating Changes to Branch

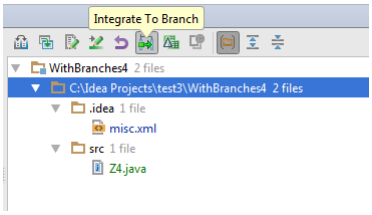
PhpStorm allows you to integrate changes into a selected branch, and commit the results of such integration to the repository. Do the following:

1. In the Version Control tool window, switch to the Repository tab .
2. Right-click the changelist you want to integrate, and select Subversion | Integrate to Branch from the context menu:



If you want to integrate separate files instead of the whole changelist, select them in the Changed Files pane and choose Subversion | Integrate To Branch from the context menu.

If you are using SVN 1.5 or higher both on the server and in your local working copy, select the relevant changelist/file(s) in the Changelists or Changed Files pane and click on the toolbar.



The Integrate to Branch dialog opens displaying the URL addresses of the source and target branches and a list of available local working copies.

3. From the Integrate into working copy list, select the path to the local working copy into which the selected changelist will be integrated.
   To add a path to the list, click the + button.

   To remove a path from the list, click the — button.

   Note Make sure the specified working copy directory is under Subversion version control!

4. To preview the merge result by enabling the `--dry-run` switch of the `svn` command, select the Try merge, but make no changes check box.
   If this check box is not selected, sources are merged silently.
5. Click OK. The Commit Changes dialog opens.
6. Review the summary, specify the necessary options, and run commit .

Viewing and Fast Processing of Changelists

If you are using Subversion 1.5 or higher on the server and in your local working copy, you can take advantage of the extended Merge Info functionality implemented through the Merge Info pane of the Version Control tool window , the Repository tab .

With this functionality, instead of browsing all changelists within a certain period, you can define a set of changelists to display. This is done by specifying a pair of branches in the repository (source and target), whereupon PhpStorm shows only the changelists from the source branch that have been integrated into the target branch.

Additionally, you can specify various filtering options to minimize the number of extraneous changelists.

Finally, integration and managing integration status are also available directly from the Merge Info pane.

The extended browsing functionality includes:

- Defining the Set of Changelists to Display
- Filtering Out Extraneous Changelists
- Viewing and Managing Integration Status
- Integrating Files and Changelists from the Version Control Tool Window

Tip Before enabling the extended Merge Info functionality, make sure you are using SVN Server 1.5 or higher.

If you are using SVN 1.4 or lower, to enable the Merge Info functionality, you need to change the format of your local working copy first.

Defining the Set of Changelists to Display

With the extended Merge Info functionality, you can limit the set of changelists to be displayed in the Changelists pane to changelists that have been integrated from one specific branch to another. These branches are referred to as "source" and "target" respectively.

To define the set of branches you want to display, do the following:

1. Open the Version Control tool window and switch to the Repository tab .
2. To open the Merge Info pane, click the Highlight Integrated button on the toolbar.
3. In the From field, specify the URL address of the source branch.
4. In the To field, specify the path to the target branch. If necessary, use the Browse button to open the Select Branch dialog.
5. Specify the path to the local working copy to which you are going to apply patches created based on the selected changelists. If necessary, use the Browse button to open the Configure Working Copy Paths dialog box.

Filtering Out Extraneous Changelists

With the extended Merge Info functionality, you can limit the number of changelists displayed in the Changelists pane by applying the following filters:

> **Tip** Filtering out integrated/not integrated changelists is available only when integration status is highlighted.
>
> Otherwise the Filter out integrated button and Filter out not integrated button are disabled. To enable integration status highlighting, click the Highlight Integrated button on the toolbar.

- To display only the changelists that have not been integrated into the working copy, click the Filter out integrated button on the toolbar.
- To display only the changelists that have been integrated into the working copy, click the Filter out not integrated button on the toolbar.
- To hide the changelists that are managed in another VCS or are located under another root, click the Filter out others button on the toolbar.
- To display only the changelists that were committed by a specific user, select User in the Filter by drop-down list. Then select the required user name.
- To display only the changelists applied to a specific module or folder, select Structure in the Filter by drop-down list. Then select the required location.
- To group changelists by users who committed them, or by commit dates, select the corresponding option in the Group by drop-down list.

Viewing and Managing Integration Status

With the extended Merge Info functionality, you can view and update changelists' integration status.

Subversion stores the information on whether a changelist has been integrated into the local working copy or not. Based on this data, PhpStorm informs you on the integration status of a specific changelist by displaying one of the following icons next to it:

- integrated
- not integrated
- integration status unknown
- common history

To have the integration status displayed, click the Highlight Integrated button on the Merge Info pane toolbar.

You can change the integration status of a changelist without actually integrating it into the working copy or reverting the previous integration. This will affect only the administrative data.

To toggle the integration status of a changelist, do one of the following:

- To mark a changelist as Integrated, select it and click the Mark As Merged button on the Merge Info pane toolbar.
- To mark a changelist as Not integrated, select it and click the Mark As Not Merged button on the Merge Info pane toolbar.

Integrating Files and Changelists from the Version Control Tool Window

You can integrate changelists or files into your local working copy directly from the Version Control tool window.

Do the following:

1. In the Changelists pane, select the required changelist. If neccessary, you can select several changelists at a time.
2. Do one of the following:
   - To integrate an entire changelist, click the Integrate to Branch button on the Merge Info pane toolbar.
   - To integrate a particular file from the selected changelist, select the file in the Changed Files pane and click the Integrate to Branch button on the Merge Info pane toolbar.
   - To revert the last integration of the selected changelist into the working copy, click the Undo Integrate to Branch button on the toolbar.

Viewing Merge Sources

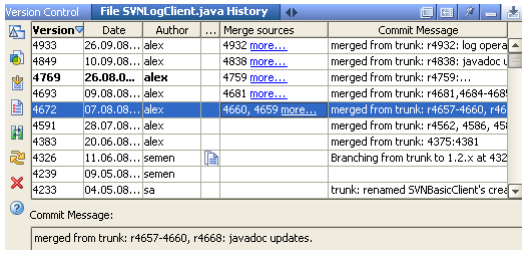
For Subversion, the common Viewing Change History functionality is expanded with the possibility to view which revisions have been used in merges applied to a file. The merge sources can be presented as a tree of revisions.

To view the merge source

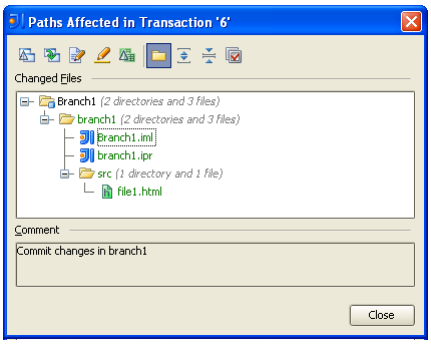
1. Open the Version Control tool window, Repository tab, and switch to the Changelists pane.

2. Select the required changelist, switch to the Changed Files pane, and select the required file.

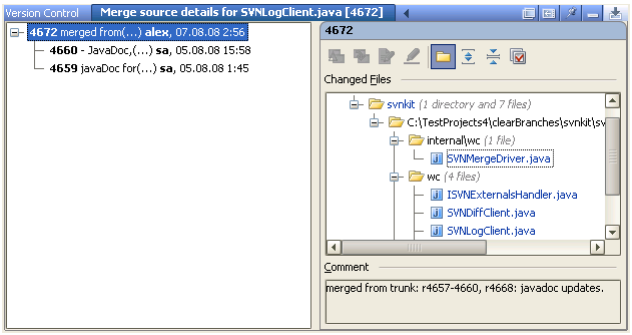
3. Select Show History in the context menu or click on the main editor toolbar. The Version Control tool window opens displaying the change history of the selected file. For each revision, the Merge sources read-only field shows the revisions that were used as sources for the merge.



4. To view all the paths affected by a merge, select the relevant row and select Show all paths affected in the selected transaction in the context menu or click on the tool window toolbar.
   The following information dialog box appears:



5. To view the source details of a merge, select the relevant row and click More. The details are displayed in the following information dialog box:



Checking SVN Project Status

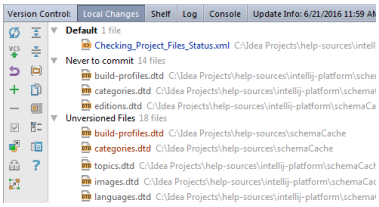
Besides indicating the current file status relative to the repository, PhpStorm integration with SVN provides you with the accumulated view of the project files' statuses.

To view differences between the current state of the project files and the repository

1. Open the required project.

2. On the main menu, choose VCS | Refresh File Status.

3. Switch to the Version Control tool window, tab Local .



The status of each file is indicated by the color in which the path to the file is displayed.

Diagnosing Problems with Subversion Integration

On this page:

- Basics
- VCS and its general settings
- Subversion settings
- The format of the local working copy
- Parent folders of the branches used
- Enabling svnkit logging

Basics

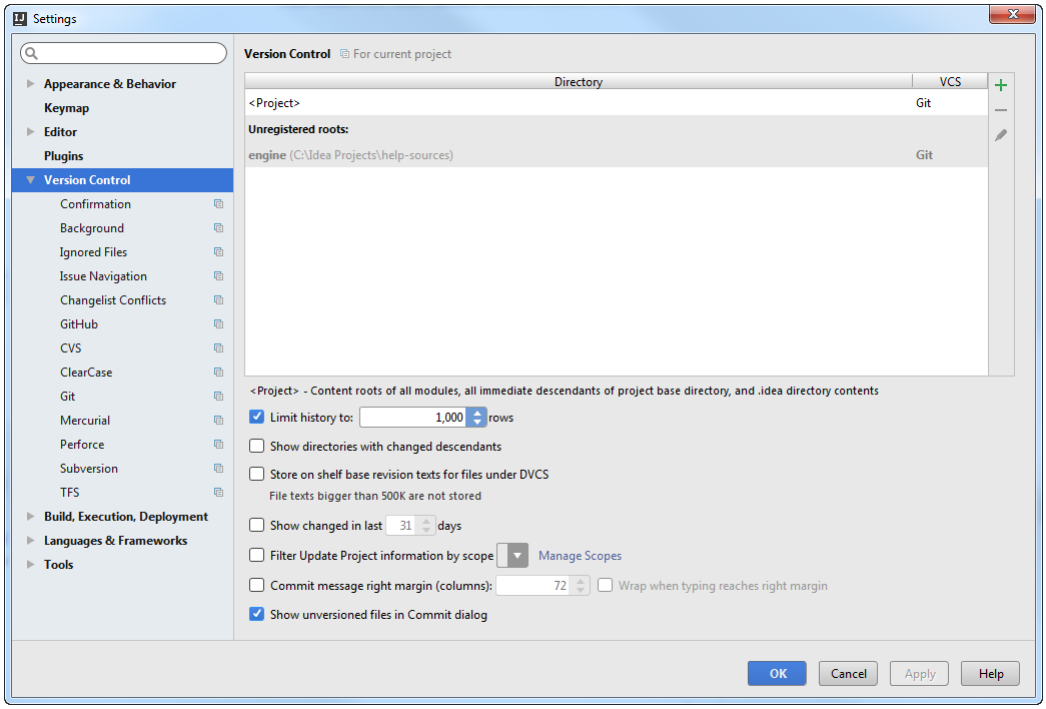
If any problem occurs with Subversion integration, feel free to contact our support. To facilitate detecting, locating, and resolving your problems, provide detailed information regarding your Subversion integration with PhpStorm. This topic lists the necessary information and explains how you can retrieve it.

The following data is usually required to diagnose Subversion problems:

- The PhpStorm version and the build number.
- The operating system used.

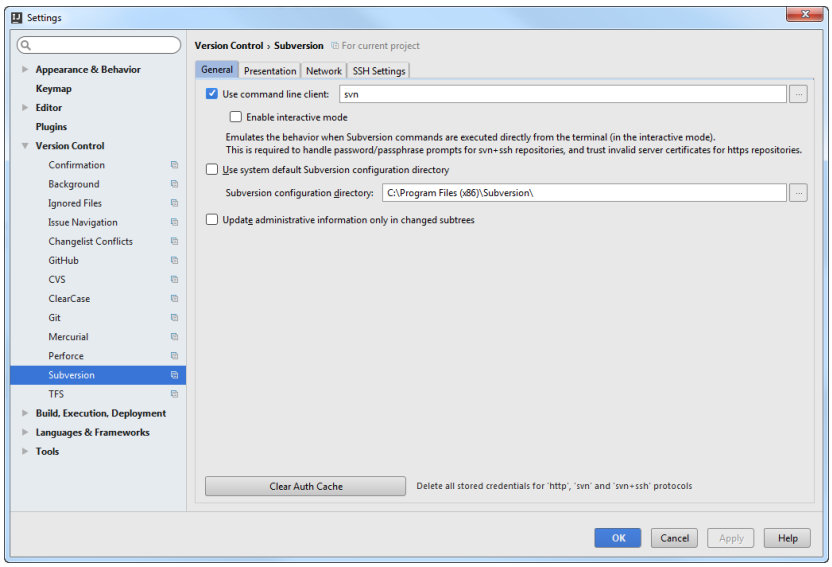
VCS and its general settings

The version control system used in your project or in its parts and the general version control settings applied to your project are specified on the Version Control page of the Settings/Preferences dialog. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Version Control.



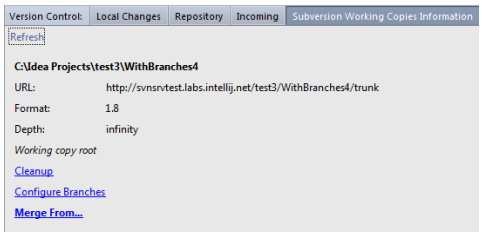
Subversion settings

Subversion-specific settings are configured on the Subversion page, under the Version Control node of the Settings/Preferences dialog. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Subversion under Version Control.



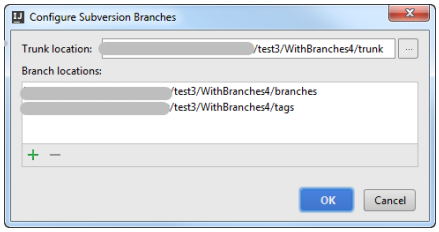
The format of the local working copy

The working copy format is the Subversion format in compliance with which the working copy was created. To view the format of a working copy, choose View | Tool Windows | Version Control on the main menu. In the Version Control tool window that opens, switch to the Subversion Working Copies Information tab.



Parent folders of the branches used

To view the configuration of branches, in the Subversion Working Copies Information tab of the Version Control tool window, click the Configure Branches link.



Enabling svnkit logging

If data saved in the PhpStorm logs is not sufficient to solve the problem and the problem is related to the communication protocol or authentication, enable svnkit logging, reproduce the problem, and attach the `idea.log`.

To enable svnkit logging, add `-Djavasvn.log=true` to one of the following files depending on the operating system used:

- Windows: `PhpStorm.exe.vmoptions`
- Linux: `PhpStorm.vmoptions`
- OS X: http://stackoverflow.com/a/13581526/72788

Refer to sections Tuning PhpStorm to learn about PhpStorm `idea.properties` and `vmoptions` files locations.

Using TFS Integration

With the TFS integration enabled, you can perform basic TFS operations from inside PhpStorm.

PhpStorm supports TFS up to TFS 2015.

The information provided in this section assumes that you are familiar with the basics of the TFS version control system.

Before you start

- The TFS integration does not require a standalone TFS client. All you need is an account in your Team Foundation Server.
- The TFS Integration plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
- Make sure that TFS integration is enabled for the entire project root or a specific directory. For more details, see Enabling Version Control .

TFS support

- When TFS integration with PhpStorm is enabled, the TFS item appears on the VCS menu, and on the context menus of the Editor and Project views.
- The files in the folders under the TFS control are highlighted according to their status. See File Status Highlights for file status highlighting conventions.
- Modifications results are shown in the Version Control tool window .
- When using TFS integration, it is helpful to open the Version Control tool window. The Console tab displays the following data:
  - All commands generated based on the settings you specify through the PhpStorm user interface.
  - Information messages concerning the results of executing generated TFS commands.
  - Error messages.
- TFS integration with PhpStorm allows you to resolve/associate work items with the new changeset when committing your changes. To achieve this, you can access saved queries and browse work item trees.

In this section:

- Creating and Managing TFS Workspaces
- Checking Out from TFS Repository
- TFS Check-in Policies

Creating and Managing TFS Workspaces

The interaction between your TFS server and local projects is configured through workspaces . A workspace mainly maps the folders in the repository with their copies on your machine.

In PhpStorm, you can configure access to several TFS servers and have as many workspaces under them as you need. The list of available TFS server access configurations and workspaces is handled through the Manage TFS Servers and Workspaces dialog box.

In this topic:

- Opening the Manage TFS Servers and Workspaces dialog box
- Configuring access to a TFS server
- Creating a workspace

To open the Manage TFS Servers and Workspaces dialog box

1. Open the Settings dialog box , and then click TFS under the Version Control node.
2. On the TFS page that opens, click the Manage button in the Servers and Workspaces area. The Manage TFS Servers and Workspaces dialog box shows the list of all available servers and workspaces in them.

To configure access to a TFS server

1. Open the Manage TFS Servers and Workspaces dialog box with the list of all available servers and workspaces in them.
2. Click the Add button in the Team Servers area.
3. In the Add Team Foundation Server dialog box, that opens, specify the URL address of the target server in the Address text box.
   In the Auth field, specify the authentication protocol to access the server. TFS uses NTLM authentication , so native Windows applications (that is, Microsoft Team Explorer) authenticate silently with system credentials. PhpStorm users must always specify their username and password because of limitations posed by Java Runtime.

   To authenticate through OAuth (Windows Live ID), choose Alternate from the Auth drop-down list.
4. For the NTLM and Alternate authentication types, specify your credentials:
   - For NTLM: the network domain where the TFS server is located, your TFS user name, and your TFS password.
   - For Alternate: your TFS user name and password.
5. Click OK. PhpStorm returns to the Manage TFS Servers and Workspaces dialog box, where the new server is added to the list.

To discard a server access configuration, select the server in the list and click the Remove button in the Team Servers area.

To create a server workspace

A workspace is identified by its name and the name of its owner, contains the name of your machine, the URL address of the server the workspace belongs to, and a set of mappings between remote and local working folders that are accessible through the workspace.

1. Open the Manage TFS Servers and Workspaces dialog box, and select the server in question. To refresh the list of the available server workspaces, click the Reload workspaces button.
2. Click the Create button in the Workspaces area.
3. In the Create Workspace dialog box, that opens, specify the workspace name. Optionally, provide a brief description of the workspace in the Comment text box.
4. In the Working folders area, define the mappings.
   1. Click the Add button +. A new line is added to the list of mappings.
   2. In the Server path text box, specify the folder on the server you need to work with.
   3. In the Local path text box, specify the local folder to store the downloaded data in.
   4. Specify the status of the mapping in the Status drop-down list.
      - To enable retrieving data from the server according to the mapping, choose Active.
      - To prevent downloading data from the server according to the mapping, choose Cloaked.
   5. To discard a mapping, select it in the list and click the Remove button –.

Checking Out from TFS Repository

Checkout helps you obtain a local working copy of the repository folders, which you can edit as required. After making the necessary changes, you can publish results by committing, or checking in changes to the repository.

Interaction between your TFS server and the local projects is configured through workspaces that map the folders in the repository with their copies on your machine. You can have remote folders downloaded either according to the mappings from an existing workspace or define the required mappings during download and have the corresponding workspace generated automatically. In either case, the Checkout from TFS wizard is used. Upon checkout, a PhpStorm project is created around the downloaded sources.

On this page:

- Downloading data to a new workspace generated automatically
- Downloading data to an existing workspace

To have a new workspace generated

1. On the main menu, choose VCS | Checkout from Version Control | TFS. The Checkout from TFS wizard starts.

2. On the Checkout Mode page, choose the Create workspace automatically option, specify the name to identify the workspace, and click Next.

3. On the Source Server page, specify the server to download the data from. Do one of the following:
   - Choose one of the existing server access configurations.
   - Click Add and define a new server access configuration in the Add Team Foundation Server dialog box, that opens.

   Click Next, when ready.

4. On the Choose Source and Destination Paths page, define the working folder mapping to generate a workspace around:
   1. In the Source path area, select the remote folder to download.
   2. In the Destination path text box, specify the location to store the downloaded data in. Type the path manually or click the Browse button ... and select the folder in the dialog box, that opens.

   Click Next.

5. On the Summary page, check the details of the workspace to be generated and click Finish to launch the checkout procedure. When the checkout is completed, PhpStorm creates a project around the downloaded sources and suggests to open it.

To download data to an existing workspace

1. On the main menu, choose VCS | Checkout from Version Control | TFS. The Checkout from TFS wizard starts.

2. On the Checkout Mode page, choose the Choose workspace manually option and click Next.

3. On the Source Workspace page, choose the server to download the data from and the workspace of this server to add the working folders to. If necessary, create new server access configurations and workspaces .
   Click Next, when ready.

4. On the Choose Source Path page, specify the remote folder to download. Such folders are available only if the selected workspace contains a mapping either for the folder itself or for its parent. If such mapping is missing, return to the Source Workspace page and add the required mapping .
   Click Next, when ready.

5. On the Summary page, check the details of the workspace to be generated and click Finish to launch the checkout procedure. When the checkout is completed, PhpStorm creates a project around the downloaded sources and suggests to open it.

TFS Check-in Policies

A check-in policy is a rule that is executed before every check-in to ensure that the selected changeset is OK to commit. Standard policies are stored on the server and are executed on the client machines.

Custom policies are implemented as custom plugins to PhpStorm. The IDs of these plugins are stored on the server, while the policies themselves are applied locally. Therefore, to enable the use of a policy in a team, all the team members should install the corresponding plugin.

On this page:

- Defining the default policy settings
- Suppressing the default PhpStorm-wide check-in policy settings in a current project
- Managing the list of available policies
- Introducing a custom check-in policy

To define the default policy settings to be applied at the PhpStorm level

1. Open the Settings dialog box , and then click TFS under the Version Control node.
2. On the TFS page, that opens, select the applicable checkboxes in the Checkin policies compatibility area.
   - Evaluate Team Explorer policies: select this check box to have the Microsoft Team Explorer policy definitions installed and executed on the client machine.
   - Evaluate Teamprise policies: select this check box to have the Teamprise policy definitions installed and executed on the client machine.
   - Warn about not installed policies: select this check box to have warnings displayed in case the specified policy definition is not installed.

To suppress applying the default check-in policy settings to a project

1. Open the Settings dialog box , and then click TFS under the Version Control node.
2. On the TFS page, that opens, click the Manage button in the Servers and Workspaces area.
3. In the Manage TFS Servers and Workspaces dialog box, that opens, select the project in question from the Team project drop-down list.
4. In the Compatibility area, select the Override default settings for team project <project name> check box.
5. Re-define the default settings by selecting or clearing the corresponding check boxes below.
   - Evaluate Team Explorer policies: select this check box to have the Microsoft Team Explorer policy definitions installed and executed on the client machine.
   - Evaluate Teamprise policies: select this check box to have the Teamprise policy definitions installed and executed on the client machine.
   - Warn about not installed policies: select this check box to have warnings displayed in case the specified policy definition is not installed.

To manage the list of available policies

The list of available policies consists of standard third-party policies and custom, user-defined policies.

1. Open the Settings dialog box , and then click TFS under the Version Control node.
2. On the TFS page, that opens, click the Manage button in the Servers and Workspaces area.
3. In the Manage TFS Servers and Workspaces dialog box, that opens, select the required workspace and click the Checkin Policies button.
4. In the Edit Checkin Policies dialog box, that opens, configure the list of policies:
   - To activate a policy, select the Enabled check box next to it.
   - To suppress a policy, clear the Enabled check box next to it.
   - To discard a policy permanently, select it in the list and click the Remove button.

To introduce a custom check-in policy

1. Implement the required policy as a custom plugin .
2. Download, install, and enable the plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Using GitHub Integration

PhpStorm supports integration with the GitHub remote storage.

To use GitHub integration, perform these general steps

- Enable the GitHub bundled plugin to get access to GitHub integration.
- Register your GitHub account in PhpStorm.
- Clone repositories from GitHub.
- Publish your projects on GitHub.
- Share code snippets through Git gists.

Registering GitHub Account in PhpStorm

To retrieve data from GitHub repositories and share your projects in right from PhpStorm, you need to register your GitHub account credentials in the IDE. You can also create an account on the GitHub free hosting without leaving PhpStorm.

In either case, PhpStorm remembers the login and password so you do not need to specify them while retrieving or uploading data.

To register GitHub account credentials

1. Open the IDE Settings and click GitHub.
2. On the GitHub page that opens, select the type of authentication that you want to use from the Auth Type drop-down list. The following options are available:
   - Password. If this option is selected and you have two-factor authentication enabled in your GitHub account settings, you will be asked to enter an authentication code each time PhpStorm requires you to log in to your GitHub account.
   - Token (recommended by GitHub for authentication from third-party applications, as it does not require PhpStorm to remember your password).
3. Specify the credentials depending on the selected authentication type and click OK.

To create a GitHub account from PhpStorm

1. Open the IDE Settings and click GitHub.
2. On the GitHub page that opens, click the Sign up link.
3. On the Sign up for GitHub page that opens in the browser, choose the account plan (free or paid) and specify the requested information. When you are through with creating the account, the GitHub Welcome Page is displayed.
   Free accounts do not support private repositories. Private repositories are available only for you, for other users even READ access is prohibited.
4. Return to the GitHub page and specify your GitHub account credentials in the Login and Password text boxes and click OK.

Cloning a Repository from GitHub

Besides the standard procedure for cloning a remote repository, PhpStorm provides facilities to clone a repository located on the GitHub remote storage.

Tip Make sure, you have registered your GitHub account credentials in PhpStorm.

To clone a repository from the GitHub storage

1. Choose VCS | Checkout from Version Control | GitHub on the main menu. PhpStorm establishes connection with GitHub using the login and password you registered . Upon establishing connection, the Select Git Hub Repository to Clone dialog box opens.
2. From the Repository drop-down list, select the source repository to clone the data from.
3. To view the details and description of the selected repository in the browser, click the Details on <repository name> here link.
4. In the Folder text box, specify the directory where the local repository for cloned sources will be set up. Type the path to the directory manually or click the Browse button ... and choose the desired directory in the Select project destination folder dialog box that opens.
5. In the Project name text box, specify the name of the project to be created based on the cloned sources.
6. Click the Clone button to start cloning the sources from the specified remote repository.

Viewing the GitHub Version of a File

You can view the GitHub copy of a file right from the IDE. PhpStorm detects which branch is currently active and opens the GitHub copy of the file in the corresponding branch.

To view the remote copy of a file

1. Open the file in the editor or select it in the Project tool window.
2. On the context menu, choose Open in Browser. The GitHub copy of the file in the remote version of the current branch opens.

Note If the command is invoked on a folder, the page with the information on the corresponding remote folder is displayed.

Publishing a Project on GitHub

You can publish your project sources in a GitHub Repository right from PhpStorm.

Tip Make sure that you have registered your GitHub account credentials in PhpStorm.

To publish your project sources on GitHub

1. On the main menu, choose VCS | Import into Version Control | Share Project on GitHub.
   - If you have registered login and password , PhpStorm establishes connection with GitHub using these credentials.
   - If you have not registered your GitHub credentials in PhpStorm, the Login to GitHub Dialog opens. Specify your GitHub login and password or create an account there.

2. Upon establishing connection, the Share Project on GitHub dialog box opens. Specify the name of the repository to store your project sources in. By default, PhpStorm suggests the name of the current project. Provide a brief description of the project functionality.
   Click the Share button. PhpStorm initiates creation of the new repository on the GitHub and uploads the project sources to it.

Creating Gists

To share your code, you can create gists right from PhpStorm. A gist can contain a code snippet, an entire file, several files, and even folders.

You can also use gists to save and share console output during running, debugging, testing, etc.

On this page:

- Enabling creation of gists
- Saving code in a gist
- Saving console output in a gist

Enabling creation of gists

1. Make sure the GitHib plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

2. Create an account on the GitHub remote storage, and register this GitHub account in PhpStorm.

Saving code in a gist

1. Select the code to be shared by doing one of the following:
   - To have a piece of code from a single file shared, open the file in the editor and select the required code snippet. If no selection is made, the gist will contain the entire file.
   - To have the code from a file shared, select the file name in the Project view.
   - To create a gist from a folder, select the folder in the Project view.
   - To create a gist from several files or folders, select them in the Project view.

2. On the context menu of the selection, choose Create gist.

3. In the Create Gist dialog box, that opens, provide a brief description of the gist to be created.

4. Configure the gist's visibility.
   - By default, the new gist will be public, that is, visible for all registered users with your login displayed as owner. To accept this default behaviour, just click OK.
   - To create a private gist that will be available for you only, select the Private check box.
   - To make the gist available for all registered users without displaying your login, select the Anonymous check box.

5. Specify whether you want PhpStorm to open the new gist in a browser or not:
   - To have the gist opened in the default PhpStorm browser so you can edit it right now, select the Open in browser check box. When you click OK, the https://gist.github.com/ page, that opens, shows the new gist with the selection or the entire contents of the current file. By default, the gist is named after the file the gist originates from. To update the code snippet, if necessary, click Edit.
   - If you do not want to open the gist right now, clear the Open in browser check box. PhpStorm displays the URL address of the gist so you you can access it later.

Saving console output in a gist

1. During a run, debug, or test session, or when running commands in the command line mode, switch to the relevant tool window or tab that presents the output of the application or command.

2. Click the right mouse button anywhere in the tab and choose Create Gist on the context menu.

3. In the Create Gist dialog box, that opens, provide a brief description of the gist to be created and configure the gist's visibility .

4. Specify whether you want PhpStorm to open the new gist in a browser or not .

Creating a Pull Request

Creating pull requests is a nice way to share your code with the community and to follow the collaborative development. With it you can tell others about the changes you've made and ask for comments, review, or just share the knowledge. With all this going on, a pull request appears in the original repository only after approval. A pull request can be prepared right from PhpStorm without switching to the browser.

To create a pull request:

1. On the main menu, choose VCS | Git | Create Pull Request. The Create Pull Request dialog box opens.
2. In the Base Form field, specify the repository to apply the changes to. PhpStorm attempts to retrieve all the relevant repositories and shows them in the drop-down list. Choose the repository from the drop-down list or click Select Other Fork and from the Form Owner drop-down list choose the name of the user who is the owner of the target repository.
3. In the Base Branch field, specify the branch to apply the changes to. Click Show Diff to view the list of commits to be included in the pull request. To view the details of a commit, select it and switch to the Log tab which shows a list of files included in the selected commit list. See also Merging, Deleting, and Comparing Branches .
4. In the Title field, specify the name for your request.
5. In the Description field, optionally provide a brief description of the changes to be applied through the request.

Managing Plugins

In this section:

- Plugins
- Enabling and Disabling Plugins
- Installing, Updating and Uninstalling Repository Plugins
- Installing Plugin from Disk
- Managing Enterprise Plugin Repositories
- Adding Plugins to Enterprise Repositories

Plugins

On this page:

- Basics
- Categories of plugins
- Plugin repositories

Basics

Plugins are extensions to PhpStorm core functionality. They provide the IDE integration with version control systems (VCS) and application servers, add support for various development technologies, frameworks and programming languages, and so on.

The more plugins are installed and enabled, the more features you have available. On the other hand, disabling unnecessary plugins may increase the IDE performance, especially on "less powerful" computers.

Certain plugins are independent, certain aren't. Dependent plugins require other plugins to be enabled.

Categories of plugins

In relation to PhpStorm, plugins may be attributed to one of the following categories:

- Plugins bundled with the IDE. These plugins are installed and enabled by default. You can disable unnecessary bundled plugins, but you cannot uninstall them. See Enabling and Disabling Plugins .
- Repository plugins, that is, plugins stored in plugin repositories (e.g., the JetBrains Plugin Repository). To be able to use the repository plugins, you should download and install them. See Installing, Updating and Uninstalling Repository Plugins .

Plugin repositories

PhpStorm provides access to the JetBrains Plugin Repository at http://plugins.jetbrains.com and to the dedicated PhpStorm Plugin Repository. The PhpStorm Plugin Repository resides at http://plugins.jetbrains.com/phpStorm and stores numerous plugins created by the PhpStorm community members.

You can set up your own, enterprise plugin repositories, for example, to store plugins that you want to reserve for your company's internal use only. (A plugin repository corresponds to one or more Web servers.)

You can register such repositories in PhpStorm and then work with them the same way as with the JetBrains Plugin Repository. See Managing Enterprise Plugin Repositories and Adding Plugins to Enterprise Repositories .

Enabling and Disabling Plugins

1. Open the Settings dialog (e.g. `Ctrl+Alt+S`).
2. In the left-hand pane, select Plugins.
3. On the Plugins page that opens in the right-hand part of the dialog, do one of the following:
   - To enable a plugin, select the check box to the right of its name.
   - To disable a plugin, clear the corresponding check box.
4. If the plugin of interest is not present in the list (which may be the case for a repository plugin), download and install the plugin.
5. Restart PhpStorm for the changes to take effect.

Installing, Updating and Uninstalling Repository Plugins

On this page:

- Introduction
- Downloading and installing repository plugins
- Updating repository plugins
- Uninstalling repository plugins

Introduction

To be able to use repository plugins , you have to download and install such plugins first. After that, you get the ability to update these plugins when appropriate.

When a repository plugin becomes unnecessary, you can disable or uninstall it.

Downloading and installing repository plugins

To download and install a repository plugin

1. Open the Settings dialog (e.g. `Ctrl+Alt+S`).
2. In the left-hand pane, select Plugins.
3. On the Plugins page that opens in the right-hand part of the dialog, click the Install JetBrains plugin or the Browse repositories button.
4. In the dialog that opens (the Browse JetBrains Plugins Dialog or the Browse Repositories dialog), right-click the required plugin and select Download and Install.
   Note that when looking for the plugin of interest, you can filter the plugin list and also to perform a search.
5. Confirm your intention to download and install the selected plugin.
6. Click Close.
7. Click OK in the Settings dialog and restart PhpStorm for the changes to take effect.

Note that the plugin you have installed is automatically enabled. When necessary, you can disable it as described in Enabling and Disabling Plugins .

Updating repository plugins

To update a repository plugin

1. Open the Settings dialog (e.g. `Ctrl+Alt+S`).
2. In the left-hand pane, select Plugins.
   The Plugins page opens in the right-hand part of the dialog.
3. Right-click any of the plugins and select Reload List of Plugins.
   The names of repository plugins that have newer versions available are shown blue.
4. Right-click the necessary plugin and select Update Plugin.
5. Click OK in the Settings dialog and restart PhpStorm for the changes to take effect.

Uninstalling repository plugins

To uninstall a repository plugin

1. Open the Settings dialog (e.g. `Ctrl+Alt+S`).
2. In the left-hand pane, select Plugins.
3. On the Plugins page that opens in the right-hand part of the dialog, right-click the repository plugin to be uninstalled and select Uninstall.
4. Confirm your intention to uninstall the selected plugin.
5. Click OK in the Settings dialog and restart PhpStorm for the changes to take effect.

Installing Plugin from Disk

Warning! Before installation, make sure that the desired plugin has already been downloaded and saved on your computer as `.zip` or `.jar` file.

To install a plugin from the disk

1. Open the Settings/Preferences dialog box and select Plugins on the left pane.
2. On the right pane of the dialog, click the Install plugin from disk button.
3. In the dialog that opens, select the desired plugin. You can quickly locate and select the necessary file if you drag-and-drop the corresponding item from your file browser (Explorer, Finder, etc.) into the area where the tree is shown. Click OK to proceed.
4. Click Apply button of the Settings/Preferences dialog.
5. Following the system prompt that appears, restart PhpStorm to activate the installed plugin, or postpone it, at your choice.

Managing Enterprise Plugin Repositories

To be able to use plugin repositories other than the JetBrains Plugin Repository (e.g., your enterprise plugin repositories), you should specify the URLs of such repositories in PhpStorm.

To manage the list of enterprise plugin repositories

1. In the Settings dialog, click Plugins .
2. Click Browse repositories.
3. In the Browse Repositories dialog that opens, click Manage repositories.
4. Use the Custom Plugin Repositories dialog that opens, to manage the list of URLs for custom (enterprise) plugin repositories:
   - To add a repository URL, click + (`Alt+Insert`). In the Add Repository dialog, specify the repository URL and click OK. (You can use the Check Now button to make sure that the specified URL is correct: PhpStorm will try to connect to the repository.)
   - To edit a repository URL, select the URL and click (`Enter`). In the Edit Repository dialog, edit the URL and click OK.
   - To remove a URL from the list, select the URL and click — (`Alt+Delete`).

   Click OK in the Custom Plugin Repositories dialog.
5. Click Close in the Browse Repositories dialog.
6. Click OK in the Settings dialog.

As an alternative, you can specify the list of URLs for the enterprise plugin repositories in one of the following files:

- `idea.properties`
  In this case, the following line should be added to this file:

```
idea.plugin.hosts=[URL1];[URL2];...[URLn]
```

- `PhpStorm.exe.vmoptions`
  In this case, the following line should be added to this file:

```
-Didea.plugin.hosts=[URL1];[URL2];...[URLn]
```

Both files reside in the `bin` directory under the PhpStorm installation folder.

Note that a repository added via this property is not visible in the Manage Repositories dialog.

Adding Plugins to Enterprise Repositories

To be available to PhpStorm users, all enterprise repository plugins must be listed in the file `updatePlugins.xml` along with their URLs and version numbers. This file must be available at the URL specified for the corresponding repository (see Managing Enterprise Plugin Repositories).

The plugins themselves are identified by their individual URLs and thus may be located on different Web servers.

– Adding a plugin to an enterprise plugin repository

– DTD for updatePlugins.xml

To add a plugin to an enterprise plugin repository

1. Upload your plugin JAR onto a Web server.
2. Add the plugin definition to `updatePlugins.xml`. If this file doesn't yet exist, create it at the location corresponding to the repository URL.
   The plugin definition in `updatePlugins.xml` may look similar to this:

```
<plugins>
                    ...
                    <plugin id="MyPlugin" url="http://plugins.example.com:8080/myPlugin.jar" version="1.0"/>
                    ...
                    </plugins>
```

3. To publish a new version of the same plugin, upload the corresponding plugin JAR to the repository, and change the value of the `version` attribute in the plugin definition.

DTD for updatePlugins.xml

The file `updatePlugins.xml` must correspond to the following Document Type Definition (DTD):

```
<!DOCTYPE plugins [
                        <!ELEMENT
                          plugins(plugin)*>
                    <!ELEMENT
                          plugin (#PCDATA)>
                    <!ATTLIST
                          plugin id CDATA #REQUIRED url DATA #REQUIRED version CDATA #REQUIRED>]>
```

Managing Tasks and Context

In this section:

- Managing Tasks and Context
  - Basics
  - Prerequisites
  - Supported issue tracking systems
- Enabling Integration with an Issue Tracking System
- Deleting Tasks
- Opening and Creating Tasks
- Viewing Description of a Task
- Switching Between Tasks
- Saving and Clearing Contexts
- Switching Between Contexts

Basics

PhpStorm provides facilities to set up your workflow according to the issue tracking procedure accepted in your team.

You can bind your account in an issue tracker to your project, and work on it in the discourse of tasks and contexts.

- A task is an activity performed in PhpStorm. Each task is identified by a task name.
  Normally, a task correlates with an issue in your issue tracking system. This correlation is set by using the desired issue ID as the task name. When you switch between tasks, PhpStorm cleans your workspace, creates a changelist for the task, and loads a stack trace, if any.

  Alternatively, you can define a task yourself so it reflects an activity that is not registered in your issue tracker.

  Tasks are listed in the drop-down list on the toolbar. This drop-down list is available only when you have at least one opened issue from the tracker or a self-defined task. A tracker issue has the light-violet background colour until it is opened in PhpStorm. After a tracker issue is opened in PhpStorm its background colour changes to white. Self-defined tasks always have the white background colour.
- A context is a set of files opened in the editor while working on a task or independently from it. You can switch between contexts by switching between tasks associated with them. Alternatively, you can save and clear contexts independently from any tasks.

Prerequisites

Before you start working with tasks and contexts, make sure that the Task Management plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Supported issue tracking systems

Currently PhpStorm supports integration with the following issue tracking systems:

- Jira
- YouTrack
- Lighthouse
- PivotalTracker
- Redmine
- Trac
- FogBugz
- Mantis
- Generic server
- Assembla
- Sprint.ly
- GitHub

Enabling Integration with an Issue Tracking System

On this page:

- Introduction
- Enabling issue tracker's integration

Introduction

To access your issues in the tracking system right from PhpStorm, you need to integrate your issue tracker with PhpStorm. Once opened in PhpStorm, an issue is considered a newly created PhpStorm task and added to the drop-down list on the toolbar, whereupon you can switch to it from another task. In the terms of PhpStorm task, opening an issue is referred to as creating a task .

Enabling integration between PhpStorm and an issue tracking system allows you to work on your project in the discourse of tasks and contexts and thus set up your workflow in accordance with the process established in your team.

The Tasks drop-down list is available only when you have at least one opened issue from the tracker or a self-defined task.

Enabling issue tracker's integration

To enable integration with an issue tracking system

1. Access the Servers dialog box. Do one of the following:
   - Open the Settings dialog box . Below the Tasks node, click Servers.
   - In the Open Task dialog box, click .
   - On the main menu, choose Tools | Tasks&Contexts | Configure Servers

2. In the Servers dialog box, specify the following:
   - The URL address of your issue tracking server.
   - Your account credentials on the server in question. These credentials will be different for the different issue tracking systems.
   - Specify whether you want to access the server via proxy and specify the proxy settings.
   - To allow access to the specified server for other members of your team, select the Share URL check box.
   - To check whether the specified settings ensure successful connection to the server, click the Test button.

   See reference page for the detailed description of controls.

3. Configure synchronization between PhpStorm and your issue tracking system. To do so, Open the Settings dialog box , and click Tasks. In the Tasks page, configure interaction between PhpStorm and your tracker. Do one of the following:
   - To have PhpStorm synchronize with the issue tracking system in the background on a regular basis, select the Enable issue cache check box and specify the synchronization frequency and the cache size.
     No matter whether you actually request on information from your issue tracker or not, PhpStorm will connect to your issue tracking system according to the specified frequency and refresh the cached issues. The advantage of this approach is that when you need to switch to a task, the up-to-date information is already at your disposal so you do not need to wait till PhpStorm establishes connection with the tracker and retrieves the information.

     Tip This configuration is especially recommended when working with rather "slow" issue tracking systems.

   - To have PhpStorm connect to the issue tracking system only when you actually need information on issues, clear the Enable issue cache check box.

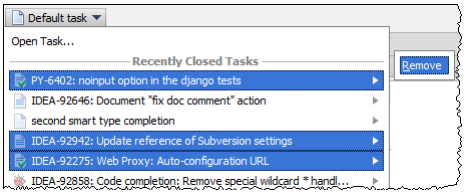
Note If a server is not trusted, PhpStorm shows a dialog box suggesting you to accept the server, or reject it. If you accept the server as trusted, PhpStorm writes its certificate to the trust store. On the next connect to the server, this dialog box will not be shown.

To delete a task

1. Click the tasks combo on the main toolbar.

2. Select one or more tasks you want to delete.
   Use `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys for multiple selection.

3. Click the right-arrow button, and choose Remove from the list:



Opening and Creating Tasks

On this page:

- Basics
- Opening tracker tasks and creating local tasks
- Tips and Tricks

Basics

PhpStorm distinguishes among tracker tasks and local tasks.

A tracker task is an issue in your tracker system until it is opened in PhpStorm whereupon it becomes a local task.

In addition to this, any activity in PhpStorm can be configured as a local task. This action is referred to as task creation.

Opening tracker tasks and creating local tasks

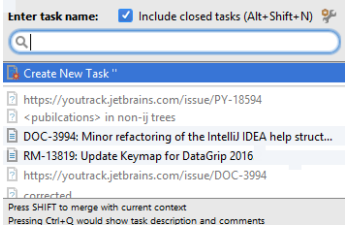
When you open a tracker task (an issue in your tracking system) for the first time, a corresponding local task is created.

To open tracker tasks and create local tasks, follow these steps

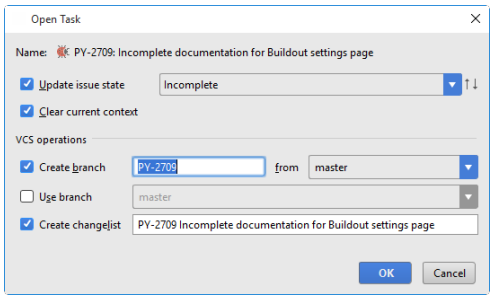
1. Do one of the following:
   - On the main menu, choose Tools | Tasks&Contexts | Open Task.
   - Press `Shift+Alt+N`
   - Click the tasks combo on the main toolbar.

2. To have the possibility to open already closed tasks, select the Include closed tasks check box. A closed task is one of the following:
   - A tracker task with the status closed in the issue tracker.
   - A local task that is not associated with a changelist provided that the entire project or the affected directory is under version control (see Associating a Project Root with a Version Control System and Associating a Directory with a Specific Version Control System).

3. In the Enter task name pop-up window, choose Create New Task, or just type the task name.



   - If you choose Create new task <description>, a new local task with the specified description will be created. In this case, the IDE will switch to this new task.
   - If you select a task from an issue tracker, then the Open Task dialog box opens.



   In this dialog, you can specify whether you want PhpStorm to clear the current context, create a changelist for the new task, change the issue status in the tracking system, create a new VCS branch or use the existing one. Refer to the dialog description for details.

Tips and Tricks

- To show task description right in the suggestion list, press `Ctrl+Q`.
- To configure access to your issue tracking system, click , and specify your account settings in the Servers dialog box that opens.
- To include closed tasks in the suggestion list, press `Shift+Alt+N` once more, or select the corresponding check box.

Viewing Description of a Task

On this page:

- Introduction
- Viewing task descriptions
- Opening tasks in browser

Introduction

When you are choosing a task to switch to , the list in the Switch to task pop-up window shows only IDs of tasks and their summaries. This information is not always sufficient, because it reflects neither the steps that lead to the problem nor the related discussion.

You can learn more about your current task in two ways:

- View the description of a task in a pop-up window.
- Open a task in the browser and view both the issue description and all the comments on it.

Viewing task descriptions

To view the description of the current task in the pop-up window

- Choose Tools | Tasks&Contexts | Show '<task ID>' Description on the main menu.

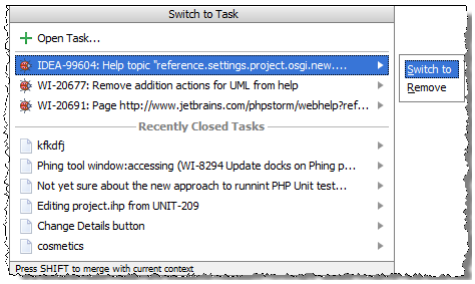
Opening tasks in browser

To open the current task in browser

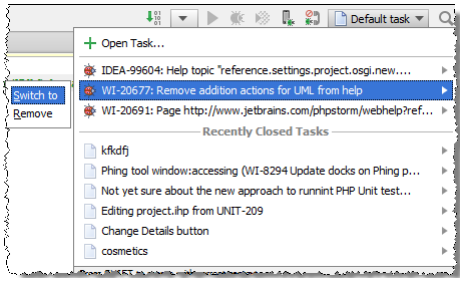
- Choose Tools | Tasks&Contexts | Open '<task ID>' in Browser on the main menu.

To switch to another task, do one of the following

- Choose Tools | Tasks & Contexts | Switch Task on the main menu. Then in the Switch to Task pop-up window, select the desired task from the list.



- Click the tasks combo in the main toolbar, select the desired task from the list, and then click Switch to on the submenu.



Saving and Clearing Contexts

With PhpStorm, you can save and clear contexts without associating them with specific tasks .

To manage the current context independently from a task, do one of the following:

- To save the current context, choose Tools | Tasks&Contexts | Save Context on the main menu.
- To clear the current context without loading another one, choose Tools | Tasks&Contexts | Clear Context on the main menu .

Switching Between Contexts

With PhpStorm, you can switch between contexts that are not associated with specific tasks .

To switch to another context

1. Choose Tools | Tasks&Contexts | Load Context on the main menu or press `Shift+Alt+L`.
2. In the Load Context pop-up window, select the desired context from the list.

Using Command Line Tools

In this section:

- Using Command Line Tools
  - Introduction
- Creating a Command Line Tool
- Running Command Line Tool Commands
- Updating a Command Line Tool

Introduction

Before you start working with Command Line tools, make sure that the Command Line Tool support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

You can run command line commands right from PhpStorm. Just configure the commands that fit your preferences and needs as a command line tool and register it in PhpStorm. You can have as many command line tools as you need.

Integration with command line tools is provided at the PhpStorm level, so once configured, a tool can be used in all your PhpStorm projects. Just activate or de-activate it when necessary depending on the needs of each specific project.

Integration with command line tools is supported via the Command Line Tool Support plugin. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

The available tools are listed in the Command Line Tool Support page of the Settings/Preferences dialog.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

Creating a Command Line Tool

On this page:

- Introduction
- Creating and configuring a command line tool
- Activating a configured command line tool

Introduction

The command line tools that are available from PhpStorm are listed in the Command Line Tool Support page of the Settings/Preferences dialog box. Integration with command line tools is provided at the PhpStorm level, so once configured, a tool can be used in all your PhpStorm projects. Just activate and de-activate when necessary depending on the needs of each specific project.

You can update the command definitions of a tool and have the `.xml` definition file validated right in the PhpStorm editor.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

Creating and configuring a command line tool

To create and configure a command line tool

1. Open the PhpStorm settings and click Command Line Tool Support.
2. In the Command Line Tool Support page, click +. In the Command Line Tools dialog box that opens, choose visibility level of the tool (project or global), and its type from the Choose tool drop-down list.
3. In the dialog box that opens, specify tool name and location, and the other required parameters.
   When you click OK, PhpStorm brings you to the Command Line Tool Support page, where the new tool is added to the list of available tools.
4. Select the newly created tool and click . In the tool definition `.xml` file that opens in the editor, define the tool commands.
5. In the Show console in area, specify where you want the Input pane for typing commands opened:
   - To have the Input pane opened in a pop-up window, choose the Pop-up option.
   - To have the Input pane opened as a text box at the bottom of the Command Line Tools Console tool window, choose the Tool window option.

Activating a configured command line tool

To activate a configured command line tool

1. Open the PhpStorm settings , and then click Command Line Tool Support.
2. Select the Enable check box next to to the tool name.

Running Command Line Tool Commands

On this page:

- Introduction
- Executing a command in the command line
- Terminating a running command
- Exporting the output
- Launching a command line tool automatically after checking in changes

Introduction

You can run commands and analyze their output right in PhpStorm using the dedicated Command Line Tools Console tool window.

The character set to be used in the tool window is chosen from the Console encoding drop-down list on the Command Line Tool Support page of the Settings dialog box. If necessary, you can export the output.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

Executing a command in the command line

To execute a command in the command line

1. To access the Input pane, choose Tools | Run Command on the main menu.
   Depending on the Show console in setting in the Command Line Tool Support page, the Input pane opens either as a pop-up window or as a text box at the bottom of the dedicated Command Line Tools Console tool window.

2. In the Input pane, type the call of the command in the format:

```
<tool alias>
                    <command>
```

   1. Besides tool-specific commands, you can also run simple, operating system-wide, commands, such as, `ls` , etc.
   2. Most command line tools operate within the context of their own projects. A tool project is not the same as a PhpStorm project. Therefore, when using a tool from PhpStorm for the first time, you need to create its own "initial" tool project besides you current PhpStorm project. To do so, run the project generation command specific for the current tool in the Command Line Tool Input pane. The "initial project" is located in the PhpStorm project root.

3. View the result of command execution in the output tab named after the last invoked command.

Terminating a running command

To terminate a running command, do one of the following:

- During the command execution, click the Stop toolbar button .
- Kill the current thread from the progress bar, if the output tab is already closed.

Exporting the output

To export the output

1. Click the Export to Text toolbar button or press `Alt+O`.

2. In the Export Preview dialog box that opens, do one of the following:
   - To have the output saved in a text file, specify the target text file in the Export to file text box and click the Save button. If the file with this name and location already exists, choose to overwrite its contents with the new data or to append the new data to the existing file.
   - To copy the output to the clipboard, click the Copy button.

Launching a command line tool automatically after checking in changes

To have PhpStorm launch a command line tool automatically after checking in changes to the repository

1. Open the PhpStorm settings and configure the required command line tool as an external tool on the External Tools page.

2. Start checking in your changes .

3. In the After Commit area, specify the tool to launch in the Run tool drop-down list. Choose one of the available tools or define a new one: click and configure the required command line tool as an external tool in the External Tools dialog box that opens.

Updating a Command Line Tool

On this page:

- Introduction
- Editing command definitions
- Reloading command definitions
- Validating the definition file for structure consistence

Introduction

You can update the command definitions of a command line tool right in the PhpStorm editor and have the `.xml` definition file validated . As you type, the file is checked for well-formedness on the fly. Full validation is performed every time you invoke a command. If any inconsistencies are detected, the tool is marked with the Invalid description icon in the Command Line Tool Support page.

You can also add new tasks to the command definition file and enable PhpStorm to use them without re-configuring the command line tool, that is, without removing it from the list of available ones and adding it anew. All you need, is reload the commands from the executable file.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

Editing command definitions

To edit the command definitions

1. Open the PhpStorm settings , and click the Command Line Tool Support.
2. In the Command Line Tool Support page, select the desired tool and click the Open definition in editor button. The corresponding `.xml` definition file opens in the editor. Update the file as necessary.

Reloading command definitions

To reload command definitions

1. Open the PhpStorm settings , and click the Command Line Tool Support.
2. In the Command Line Tool Support page, select the desired tool and click the Reload command list from executable button.

Validating the definition file for structure consistence

To validate the definition file for structure consistence

1. Start executing a command : choose Tools | Run Command on the main menu or press `Ctrl+Shift+X`. Then type the call of the command in the Input tab of the Command Line Tools Console tool window.
2. View the results in the Tool definition file errors tab. As soon as PhpStorm detects any structure inconsistency, it displays an error message that contains a brief description of the problem and indicates the line number where the problem is detected.
   By default, the tab is not displayed and opens only upon validation failure. In this case, PhpStorm displays a Command Line Tool pop-up window with a notification on validation failure. Upon clicking the More link, the Tool definition file errors tab opens showing messages on detected inconsistencies. Each message contains information on the file and the line number where the problem was found, as well as a brief description of the error.

   You can close the tab by clicking the cross on its header. To re-open the tab, again click More in the Command Line Tool notification pop-up window, which remains on the screen until you close it manually.

Using File Watchers

On this page:

- Basics
- Specifying the File Watcher name, type, and description
- Configuring the behaviour of the File Watcher
- Configuring interaction with the transpiler
- Enabling and disabling File Watchers
- Examples of customizing the behaviour of a transpiler

Basics

PhpStorm supports integration with various third-party transpilers that run in the background and perform the following:

- Translate Less, Sass, and SCSS source code into CSS code.
- Translate TypeScript and CoffeeScript source code into JavaScript code, possibly also creating source maps to enable debugging.
- Compress JavaScript and CSS code.

Note that PhpStorm does not contain built-in transpilers but only supports integration with the tools that you have to download and install outside PhpStorm.

In PhpStorm, these transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

You can download a transpiler of your choice and set it up as a File Watcher. However, in this case no predefined template is available so you will have to specify all the settings manually.

To be applicable, a File Watcher must be enabled by selecting the check box next to it on the File Watchers page of the Settings dialog box, see Enabling and Disabling File Watchers . After that the File Watcher will be invoked automatically upon any changes made to the source code or upon save, depending on whether the Immediate File Synchronization check box is selected or cleared, see New Watcher Dialog .

The output of a File Watcher is stored in a separate file. The predefined templates suggest the type of the file depending on the transpiler type. By default the output file is created in the same folder as the input file when the File Watcher is invoked for the first time, whereupon this file is only updated. You can customize all these settings during File Watcher creation.

JavaScript files generated by File Watchers are excluded from code completion and refactoring.

In the Project tree view, the output file is shown under the original file which is shown as a node. This is done to improve visibility so you can easier locate necessary files.

File watchers have two dedicated code inspections:

- The File watcher available inspection is invoked in every file that is recognized as subject for applying a predefined file watcher (Sass, Less, SCSS, TypeScript, or CoffeeScript). If none of the applicable predefined File Watchers is associated with the current project, PhpStorm suggests to add one.
- The File watcher problems inspection is invoked by a running File Watcher and highlights errors specific for it.

To create a project File Watcher based on a predefined template, perform these general steps:

1. Make sure the transpiler to use is downloaded and installed on your machine.
2. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Specify the File Watcher name, type, and description.
4. In the Options area, configure the behaviour of the File Watcher: the events that will invoke it, from which files it can be invoked, when you want the console shown, etc.
5. In the Watcher Settings area, configure interaction with the transpiler and its behaviour: the path to the executable file, the arguments to pass to it, the acceptable input and expected output file types, etc.

Specifying the File Watcher name, type, and description

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click File Watchers under Tools.

2. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project. Click the Add button + or press `Alt+Insert` and do one of the following:
   - Choose the predefined template to create the File Watcher from. The choice depends on the transpiler you are going to use.
   - To set up a transpiler of your choice, choose Custom.

   After you choose the relevant template or the Custom option, the New Watcher dialog box opens.

3. In the Name text box, type the name of the File Watcher. By default, PhpStorm suggests the name of the selected predefined template.

4. Provide a brief description of the File Watcher's functionality. Predefined templates already contain general information regarding the accepted input and produced output. Expand or update the description, if, for example, you have two File Watcher that apply to the same source code but differ from each other in some configuration settings. .

Configuring the behaviour of the File Watcher

In the Options area of the New Watcher dialog box, configure the behaviour of the File Watcher: the events that will invoke it, from which files it can be invoked, when you want the console shown, etc.

1. Specify whether you want the File Watcher to interact with the PhpStorm syntax parser:
   - To have the File Watcher ignore update, save, and change focus events in files that are syntactically invalid and start only in error-free files, select the Trigger watcher regardless of syntax errors check box.

- To have the File Watcher start regardless of the syntactical correctness of a file, clear the Trigger watcher regardless of syntax errors check box. The File Watcher will start upon update, save, or frame deactivation, depending on the status of the Immediate file synchronization check box.

2. Specify the events which will invoke the File Watcher:
   - To have the File Watcher invoked as soon as any changes are made to the source code, select the Immediate file synchronization check box.
   - Clear the check box to have the File Watcher started upon save (File | Save All)or when you move focus from PhpStorm (upon frame deactivation).

3. Specify how you want the File Watcher to deal with dependencies. When the File Watcher is invoked on a file, PhpStorm detects all the files in which this file is included. For each of these files, in its turn, PhpStorm again detects the files into which it is included. This operation is repeated recursively until PhpStorm reaches the files that are not included anywhere within the specified scope . These files are referred to as root files (do not confuse with content roots).
   - To run the File Watcher only against the root files, select the Track only root files check box.
     Note that in the TypeScript file watcher, the check box has been renamed to Compile main files only.
   - Clear the check box to run the File Watcher against the file from which it is invoked and against all the files in which this file is recursively included within the specified scope.

   This option is available only for Babel, Closure Compiler, Compass, Jade, Less, Sass/ SCSS, Stylus, Typescript, UglifyJS, and YUI Compressor JS.

4. In the Show console drop-down list, specify when you want the console shown. The available options are:
   - Always: when this option is chosen, the File Watcher opens the console when it starts.
   - Error: when this option is chosen, the File Watcher opens the console only if any errors occur in transpilation.
   - Never: choose this option to suppress opening the console under any circumstances.

5. Configure the output filters to distinguish the output of the File Watcher from other output. These filters make the basis for:
   1. Displaying paths to the File Watcher output files as links in error and other messages and logs. When you click such link, the corresponding file is opened in the editor.
   2. Error highlighting in the output files.

   Click the Output Filters button to open the Output Filters dialog and manage the list of filters.

Configuring interaction with the transpiler

In the Watcher Settings area, configure interaction with the transpiler: specify the path to the executable file, the arguments to pass to it, the acceptable input and expected output file types, etc.

1. In the Program text box, specify the location of the transpiler's executable file (`.exe` , `.cmd` , `.bat` or other depending on the specific tool. `.jar` archives are also acceptable but defining `PATH` variables for them is not supported. Do one of the following:
   - Type the path explicitly.
   - Click the Browse button to open the Select Path dialog box and navigate to the desired location.
   - Click the Insert Macro button to open the Macros dialog box where you can select the relevant macro from the list.

2. From the File type drop-down, choose the expected type of input files. The File Watcher will consider only files of this type as subject for analyzing and processing. File types are recognised based on associations between file types and file extensions .
   By default, the field shows the file type in accordance with the chosen predefined File Watcher template.

3. In the Scope drop-down list, define the range of files the File Watcher can be applied to. Changes in these files will invoke the File Watcher either immediately or upon save or frame deactivation, depending on the status of the Immediate file synchronization check box.
   You can choose one of the predefined scopes from the drop-down list:
   - Project Files: all the files within the project content roots (see Content Root and Configuring Content Roots).
   - Project Production Files: all the files within the project content roots excluding test sources.
   - Project Test Files: all the files within the project test source roots.
   - Open Files: all the files that are currently opened in the editor.

   VCS Scopes: these scopes are only available if your project is under version control .
   - Changed Files: all changed files, that is, all files associated with all existing changelists .
   - Default: all the files associated with the changelist `Default` .

   Alternatively, click the Browse button and configure a custom scope in the Scopes dialog box that opens.

   For more details on scopes, see the pages Scopes and Scopes dialog .

   The Scope setting overrides the Track only root files check box setting: if a dependency is outside the specified scope, the File Watcher is not applied to it.

4. In the Arguments text box, define the arguments to pass to the transpiler Among other cases, use this text box to change the default output location, that is, specify a custom location where you want the transpiler to store the files generated during transpilation. Do one of the following:
   - Type the list of arguments in the text box.
   - Click the Insert Macro button to open the Macros dialog box where you can select the desired macro from the list.

   When specifying the arguments, follow these rules:
   - Use spaces to separate individual arguments.
   - If an argument includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"` .
   - If an argument includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value"\` .

5. In the Working Directory text box, specify the directory to which the transpiler will be applied. Because the tool is always invoked in the context of a file, the default working directory is the directory of the current file. This setting is specified in all predefined templates through a macros `$FileDir$` . To update this default settings, do one of the following:
   - Type the path explicitly in the text box.
   - Click the Browse button to open the Select Path dialog box and navigate to the desired location.
   - Click the Insert Macro button to open the Macros dialog box, where you can select the desired macro from the list.

> **Tip** If the field is left blank, PhpStorm uses the directory of the file where the File Watcher is invoked.

6. In the Output paths to refresh text box, specify the files where the transpiler stores its output: the resulting source code, source maps, and dependencies. Based on this settings, PhpStorm detects files generated through transpilation.
   Please note, that changing the value in this text box does not make the transpiler store its output in another location. To do that, specify the desired output location in the Arguments text box.

   Do one of the following:
   - Type the output paths manually. Use colons as separators.
   - Click the Insert Macro button to open the Macros dialog box, where you can select the desired pattern from the list.

7. Use the Create output file from stdout check box to specify how you want the output file generated.
   - To have PhpStorm read the native transpiler's output (`standard output stream (stdout)`) and generate the resulting files from it, select the Create output file from stdout check box.
   - To have the transpiler write its output directly to the files specified in the Output paths to refresh field, clear the check box.

> **Tip** Some transpilers generate a `standard output stream (stdout)` file, others do not, which may lead to errors. Therefore it is strongly recommended to preserve the default setting.

8. Optionally, define environment variables. For example, specify the `PATH` variable for the tool that is required for starting the transpiler but is not referenced in the path to it. In most cases it is Node.js or ruby.exe. Such situation may happen if you installed the transpiler manually but not through the Node Package Manager or gem manager and its installation folder is not under Node.js or ruby.

Enabling and disabling File Watchers

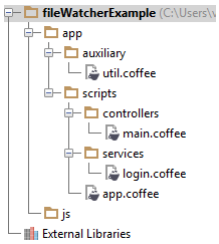
To toggle the enable/disable status of a File Watcher, select/clear the check box next to it on the File Watchers page of the Settings dialog box. If an error occurs while a File Watcher is running, the File Watcher is automatically disabled. To restore the status, enable the File Watcher manually.

When a File Watcher is enabled, it starts automatically as soon as a file to which transpilation is applicable is changed or saved, see Configuring the behaviour of the File Watcher .

Examples of customizing the behaviour of a transpiler

Any transpiler is an external, third-party tool. Therefore the only way to influence a transpiler is pass arguments to it just as if you were working in the command line mode. These arguments are specific for each tool. Below are two examples of customizing the default output location for the CoffeeScript transpiler.

Suppose, you have a project with the following folder structure:



By default, the generated files will be stored in the folder where the original file is. You can change this default location and have the generated files stored in the `js` folder. Moreover, you can have them stored in a flat list or arranged in the folder structure that repeats the original structure under the `app` node.

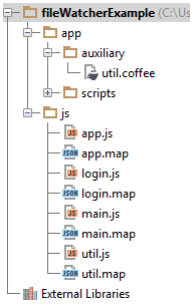
- To have all the generated files stored in the output `js` folder without retaining the original folder structure under the `app` folder:
  1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\ --compile --map $FileName$
```

  2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileNameWithoutExtension$.map
```

  As a result, the project tree looks as follows:



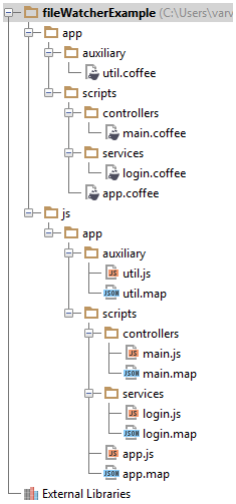
- To have the original folder structure under the `app` node retained in the output `js` folder:
  1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\ --compile --map $FileName$
```

2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileDirRelativeTo
```

As a result, the project tree looks as follows:



Working with Embedded Local Terminal

On this page:

- Prerequisites
- Overview
- Configuring embedded local terminal
- Running embedded local terminal
- Actions available in the embedded local terminal

Prerequisites

Before you start working with terminal, make sure that the Terminal plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Overview

PhpStorm features a local terminal that makes it possible to access the command line. Depending on your platform, you can work with command line prompt, Far, `powershell` , `bash` , etc. Using the terminal, you can execute any command without leaving the IDE.

Configuring embedded local terminal

The terminal settings are configurable on several pages of the Settings/Preferences dialog.

To configure the embedded local terminal options

1. In the Settings/Preferences dialog box, open the Terminal page, and specify the following:
   - The desired shell that will run by default, the name of a session tab
   - Name of the tab a new session will be opened in, possibility to copy to clipboard etc.
   - Ability to override the PhpStorm keymap.

2. The following settings are inherited by the embedded local terminal from the IDE settings/preferences:
   - On the Keymap page, you can configure the `Ctrl+C` and `Ctrl+V` shortcuts.
   - On the editor's Appearance page - anti-aliasing and caret blinking.

     Note that the setting Use block caret is not inherited in the terminal - its caret is always block.
   - Under the editor's Color and Fonts settings, you can change the following options:
     - On the Console Fonts page - line spacing, and console fonts.
     - On the Console Colors page - console colors.
     - On the General page - selection foreground and background colors.

Running embedded local terminal

To run the console, do one of the following

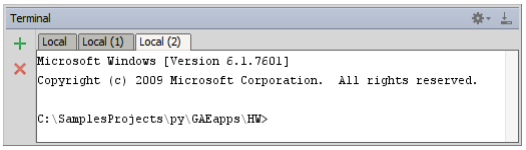
- Press `Alt+F12`.
- Click the Terminal tool window button Terminal .
- Hover your mouse pointer over in the lower left corner of the IDE, then choose Terminal from the menu, as described in the section Tool window quick access .

Actions available in the embedded local terminal

In the embedded local terminal, you can do the following:

- To create a new session:
  - Click + on the toolbar of the terminal.
  - Right-click a session tab, and then choose Create new session on the context menu.

  A new session is presented in a separate tab:



- Rename a tab. Right-click a tab, and choose Rename tab on the context menu.
- Close an active session that currently has the focus. This can be done in a number of ways:
  - Click ✕ on the terminal toolbar.
  - Right-click a session tab, and then choose Close session on the context menu.
- Use up and down arrows on your keyboard to browse through the history of entered commands.
- Toggle between the embedded local terminal and editor by pressing `Alt+F12` (View | Tool Windows | Terminal).

Running SSH Terminal

On this page:

- Introduction
- Preparing to work in the SSH Terminal
- Launching an SSH Terminal

Introduction

You can launch an SSH Session right from PhpStorm. By running commands in a dedicated SSH terminal, you can access data on a remote Web server or a Vagrant instance (virtual machine) via an SSH tunnel, mainly upload and download files.

Preparing to work in the SSH Terminal

1. Make sure the SSH Remote Run plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
2. Make sure, an SSH server is available in the destination environment: a remote Web server or a Vagrant instance (virtual machine).
3. Register an account on the SSH server in the destination environment and generate a pair of SSH keys or a password, depending on the server policy.
4. Appoint the destination environment and specify the settings to establish connection with it:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click SSH Terminal under Tools. The SSH Terminal page opens.
   2. In the Connection settings area, appoint the destination environment:
      - Current Vagrant: select this option to have the commands in the SSH Terminal executed on the currently running Vagrant virtual machine. For details, see Vagrant: Working with Reproducible Development Environments .
      - Deployment server: select this option to have the commands in the SSH Terminal executed on the local or remote Web server accessible through one of the server access configurations . From the drop-down list, choose the server access configuration that specifies the destination environment and the settings to establish connection to it.
        - Select server on every run: if this option is selected, you will have to choose the desired server access configuration from the pop-up window, every time you choose Tools | Start SSH Session on the main menu.
        - If the desired server access configuration does not appear in the drop-down list, click the link Configure Servers, and define one in the Deployment page. For details, see Configuring Synchronization with a Web Server .
   3. From the Default encoding drop-down list, select the desired encoding to be used in the SSH terminal.

Launching an SSH Terminal

1. On the main menu, choose Tools | Start SSH Session.
2. Depending on the connection settings, defined in the SSH Terminal page of the Settings dialog, the following types of behavior are possible:
   - If the Current Vagrant option has been selected, the SSH Terminal will provide access to the currently running Vagrant virtual machine. For details, see Vagrant: Working with Reproducible Development Environments .
   - If the option Deployment server has been selected, the SSH Terminal will provide control over the data on the local or remote Web server accessible through the server access configuration chosen from the list. For details, see Configuring Synchronization with a Web Server .
   - If the option Select server on every launch has been selected, PhpStorm will show a pop-up list to choose the desired server access configuration from.

Working with PhpStorm Features from Command Line

Besides working from within PhpStorm, it is possible to perform certain actions "offline", without actually launching the IDE. This way you can:

- Launching a code inspection from the command line
  - Examples
    - Windows
    - OS X
- Viewing differences
- Opening a file in the editor

Launching a code inspection from the command line

To launch a code inspection from the command line

- Specify the following command line arguments:
  - Path to the launcher: specify the full path to one of the following launchers (which reside under the `bin` directory of your PhpStorm installation):
    - For Windows: `inspect.bat`
    - For UNIX and OS X: `inspect.sh`
  - Project file path is the full path to the directory that contains the project to be inspected.
  - Inspection profile path is the full path to the profile, against which the project should be inspected. The inspection profiles are stored under `USER_HOME\.PhpStormXX\config\inspection`
  - Output path is the full path to an existing directory where the report will be stored.
  - Options. You can specify:
    - The directory to be inspected `-d <full path to the subdirectory>`
    - The verbosity level of output `-vX`, where X is 0 for quiet, 1 for noisy and 2 for extra noisy.

Warning! Please note that you have to specify full paths. Relative paths are not accepted!

Examples

Windows

```
C:\Program Files (x86)\JetBrains\<PhpStorm home>\bin\inspect.bat
E:\SampleProjects\MetersToInchesConverter E:\Work\MyProject\.idea\inspectionProfiles\Project_Default.xml
E:\Work\MyProject\inspection-results-dir -v2 -d E:\SampleProjects\MetersToInchesConverter\subdirectory
```

OS X

```
/Applications/PhpStorm.app/Contents/bin/inspect.sh ~/PhpStormProjects/MyTestProject
~/Library/Preferences/PhpStorm<version>XX/inspection/Default.xml ~/PhpStormProjects/MyTestProject/results-dir -v2
```

Viewing differences

To view differences using command line diff tool

- In the command line, type the following:

```
<PhpStorm> diff <path1> <path2>
```

where:
- <PhpStorm> is the platform-specific product launcher
- `<path1>, <path2>` are full paths to the files to be compared.

Examples

Windows

```
PhpStorm.exe diff C:\SamplesProjects\MetersToInchesConverter\src\javascript\numbers.js
                 C:\SamplesProjects\MetersToInchesConverter\src\coffeescript\numbers.coffee
```

OS X

```
/Applications/PhpStorm.app/Contents/MacOS/PhpStorm diff ~/Documents/file1.txt ~/Documents/file2.txt
```

Opening a file in the editor

To open a file for editing

- In the command line, type the following:

```
<PhpStorm> <path1> --line <number> <path2>
```

where:

- `<PhpStorm>` is the platform-specific product launcher
- `<path1>` is the path to the project that contains the desired file
- `<number>` is the number of the line, where the caret should rest
- `<path2>` is the path to the file to be opened

Examples

Windows

```
PhpStorm.exe C:\SamplesProjects\MetersToInchesConverter --line 3
C:\SamplesProjects\MetersToInchesConverter\src\javascript\numbers.js
```

OS X

```
/Applications/PhpStorm.app/Contents/MacOS/phpstorm ~/PhpStormProjects/untitled45 --line 1 ~/PhpStormProjects/untitled45/samp
```

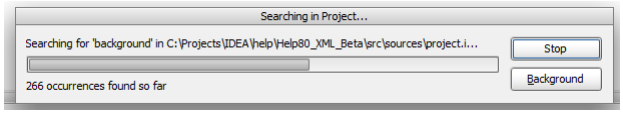
Working with Background Tasks

On this page:

- Introduction
- Viewing background tasks

Introduction

When lengthy tasks are running, for example, search and replace, VCS update, etc., PhpStorm displays the progress bar. You can bring execution of such tasks to the background by clicking the Background button.

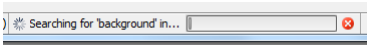


However, when a task of a certain type is brought to the background, you can still view its progress.

Viewing background tasks

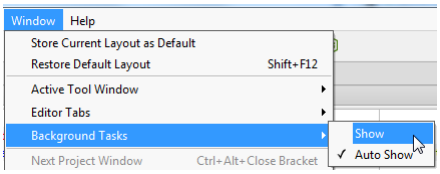
To view tasks running in the background, do one of the following

- View the Status bar:

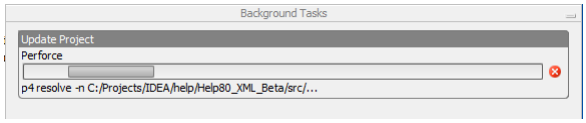


- In the Status bar , click .

- On the main menu, choose Window | Background Tasks | Show.



As a result, the Background Tasks manager opens, showing all the tasks that are currently running in the background. You can review their progress, and if necessary, cancel the unnecessary tasks by clicking .



When the Auto Show command is checked, the Background Tasks manager opens automatically every time a task is launched in the background.

Using Language Injections

In this section:

- Using Language Injections
  - Introduction
  - Prerequisites
  - Injecting a language into the source code and cancelling language injection
  - Moving injections between the IDE and the project levels
- Opening and Closing Language Injections in the Editor

Introduction

PhpStorm makes it possible to work with islands of different languages embedded in the source code. You can inject other languages into string literals. This can be done within the source code written in most (but not all) of the supported languages . The typical examples are HTML fragments injected into JavaScript code, SQL statements in XML, and so on.

PhpStorm supports full coding assistance for:

- CSS and JavaScript in an HTML or XML file.
- CSS, JavaScript, and SQL outside PHP code blocks and inside PHP string literals.
- sprintf format references in SQL injected strings.

When the editor recognizes a string as a language injection:

- Syntax and error highlighting and coding assistance are extended to this string.
- You can open and modify it in a separate tab in the editor , as if you were working with the source code in the corresponding language.
  To open an injection in the editor, use the Edit <Language> Fragment intention action .

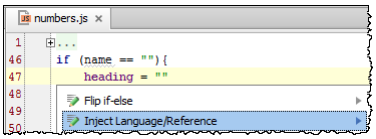
Prerequisites

Before you start working with language injections, make sure that the IntelliLang plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

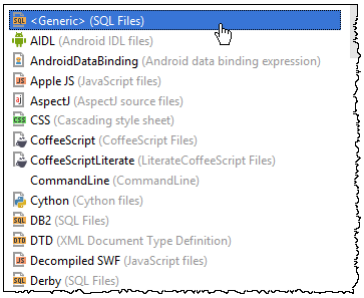
Injecting a language into the source code and cancelling language injection

To tell PhpStorm that certain text should be treated as an embedded source code fragment, you can use:

- The Inject Language/Reference intention action , which is applied to a particular string:



After that, you have to select the desired language from the menu.



Note that if you use this method, it's possible that the string literal will stay marked as a language injection only within a limited period of time. That is, PhpStorm, at a certain moment, may "forget" that the corresponding literal is a language injection. The period of the injection "persistence" will depend on the language, context and the modifications that you make in other parts of your source code.

- The Language Injections page of the Settings dialog . By creating new language injection configurations on that page, you can specify that a certain method parameter, text in an XML tag, or XML attribute should always be treated as an embedded source code fragment in another language.

> **Warning!** PhpStorm comes with a set of predefined injection configurations which is quite sufficient to ensure high productivity and comfortable environment. Therefore it is strongly recommended that you use the predefined injection configurations and avoid creating new ones.

For example, SQL injection in JavaScript code will look as follows:

```
var i;
                    for (var j=1;j<10;j++){
                    var s= "SELECT * FROM A WHERE
                    B=C";
                    }
```

To cancel language injection, choose the intention action Un-inject Language/Reference.

Moving injections between the IDE and the project levels

User-defined injections are assigned either to the IDE or the project level. The IDE-level injections are available in all of your projects. The project-level injections can be used only in the project in which they are defined.

To move the injections between the IDE and the project levels, use the Move to Project/IDE Scope icon on the Language Injections page of the Settings/Preferences dialog .

The built-in injections are the IDE-level ones, and their scope cannot be changed.

Opening and Closing Language Injections in the Editor

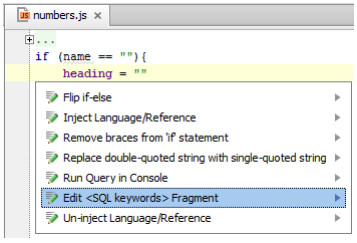
Language injections are opened in the editor by means of a dedicated intention action Open <Injected Language> Fragment.

To open a language injection in the editor:

1. In the editor, place the cursor within the string literal that represents the language injection of interest.

2. To show the available intention actions, do one of the following:
   - Click the yellow bulb.
   - Press `Alt+Enter`.

3. To select the Edit <Injected Language> Fragment option, do one of the following:
   - Click this option.
   - Use the `Up` and `Down` arrow keys to navigate to the option, and then press `Enter` to select it.



Tip You can open a number of language injections in separate editors, and copy code fragments from one editor to another. However, as soon as you start modifying the code in one of the editors, all the rest of the editors close automatically.

To close an editor for a language injection , you can use the same options as for closing files .

Comparing Files and Folders

PhpStorm helps explore differences in the various situations: differences between files, directories, revisions of the same file under version control or in the local history, database objects, local and remote files.

All these operations are performed in a similar way. In this section we'll consider the most basic operations:

- Comparing Files
- Comparing Folders

Comparing Files

On this page:

- Introduction
- Comparing two files
- Comparing a file in the editor with the Clipboard contents
- Comparing a file with the editor contents

Introduction

PhpStorm enables you to compare arbitrary files in project (including image files), selected file with the editor, or compare a file in the editor with the Clipboard contents. All comparisons are performed in the Differences viewer .

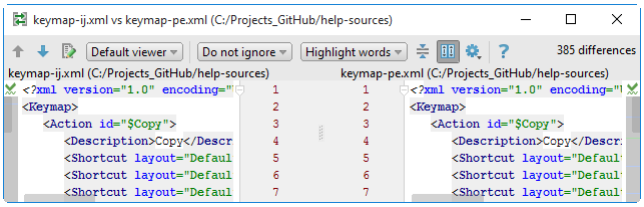
Comparing two files

1. Press and keep holding `Ctrl` for Windows and Linux or `⌘` for OS X and click the two files to compare in the Project tool window.
2. On the context menu of the selection, choose Compare Files, or press `Ctrl+D`. The Differences Viewer for Files opens, with the differences being color-highlighted.

   Tip It is enough to select a single file in the Project tool window. In this case the context menu command is Compare File with Editor, and the Differences Viewer shows the contents of the selected file on the left pane, and the contents of the active editor tab on the right pane.

3. View the differences and apply them, if necessary, using the chevron buttons » «.
   Note that keeping `Ctrl` (for Windows and Linux) or `⌘` (for OS X) pressed turns the chevron buttons » « to ⇥ ⇤. Click these buttons to append changes.

   Keeping `Shift` pressed turns the chevron buttons » « to ×. Click this button to revert changes.



Comparing a file in the editor with the Clipboard contents

1. Open the desired file in the editor.
2. Right-click the editor pane and choose Compare with Clipboard on the context menu.
3. View and manage differences in the Differences Viewer for Files .

Comparing a file with the editor contents

1. Right-click the desired file in the Project tool window.
2. Choose Compare File with Editor on the context menu.
3. View and manage differences in the Differences Viewer for Files .

Comparing Folders

On this page:

- Basics
- Opening the Difference Viewer
- Comparing two folders in the Difference Viewer
- Synchronizing contents of folders

Basics

PhpStorm provides a dedicated Differences Viewer for Folders for comparing files in two folders against the file size, content, or timestamp. The Differences Viewer shows the contents of the selected directories in the left and right panes of the Item List. The contents of the selected file are shown in the lower pane, with the differences being color-highlighted.

Besides exploring differences, the tool also provides interface for synchronizing the contents of folders.

Opening the Difference Viewer

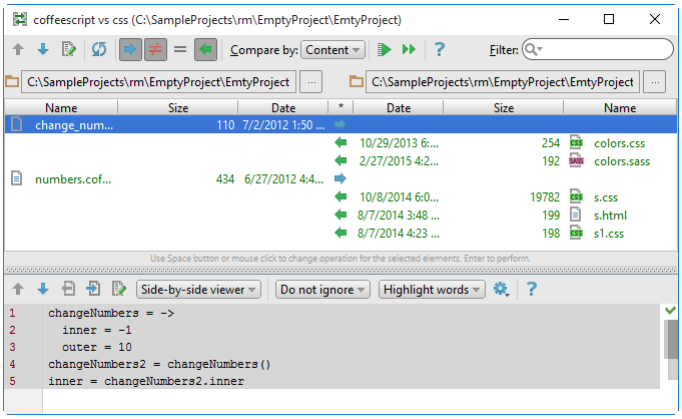
Do one of the following:

- Keeping the `Ctrl` key pressed, click two directories in the Project tool window, and choose Compare Directories on the context menu of the selection, or press `Ctrl+D`.
- Select a directory in the Project tool window, choose Compare with on the context menu of the selection, or press `Ctrl+D`, and then select the second directory in the dialog that opens .

> **Tip** You can also open the difference viewer without running PhpStorm. This is done through the following command: `<path to PhpStorm executable file> diff <path_1> <path_2>` where `path_1` and `path_2` are paths to the folders in question.

Comparing two folders in the Difference Viewer

1. Configure the layout of the Items List. Use the toolbar buttons to narrow down or widen the set of items to show. For example, show or hide files that exist in just one of the directories, equal files, or different files, etc.
2. Specify the parameter for comparison. In the Compare by drop-down list, select one of the possible options (contents, size, or time stamp).
3. Filter the folders' contents. To do that, type filtering string in the Filter text field, and press `Enter` to apply it. Using the asterisk `*` wildcard to represent any number of characters is welcome.
4. To switch to another pair of folders to compare, update the fully qualified paths to them. Click the Browse button ... next to the Paths read-only fields and choose the required folders in the dialog box that opens .
5. Explore the detected differences between files in the Differences Pane.



Synchronizing contents of folders

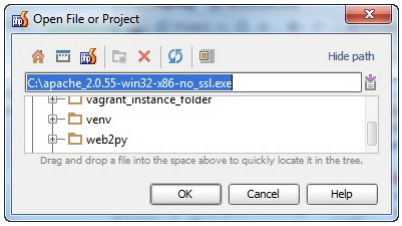
1. For each pair of items, in the * field specify the action to apply. Click the icon in the field until the required action is set.

Icon	Action
	Copy the item in the left side to the right side, possibly overwriting the contents of the corresponding target item, if it already exists.
	Copy the item in the right side to the left side, possibly overwriting the contents of the corresponding target item, if it already exists.
	The items are treated identical with regard to the selected criterion of comparison. No action will be performed by default.
	The items differ with regard to the selected criterion of comparison. No action will be performed by default. Explore the differences in the Differences Pane and change the intended action by clicking the icon.
	The item is present only in one of the folders and will be removed.

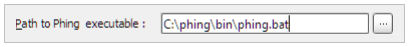
2. Do one of the following:
   - To synchronize the currently selected item, click the Synchronize Selected button on the toolbar.
   - To synchronize all the items, click the Synchronize All button on the toolbar.

Selecting Files and Folders

For selecting files and folders on your computer, PhpStorm provides a dedicated dialog whose functions are similar to those of a file browser.



In many situations, this dialog is a useful alternative to typing a path to a file or folder. All the fields in PhpStorm that contain path information have an ellipsis button ... next to them that provides access to the file chooser dialog.



The keyboard equivalent of the ellipsis button is `Shift+Enter`.

Copy and Paste Between PhpStorm and Explorer/Finder

PhpStorm enables tight interaction with the native file managers (Explorer on Windows, or Finder on Mac), and allows you exchange files and directories via the system clipboard, using numerous techniques:

- Cut, copy and paste files and directories from the Project tool window of PhpStorm to a directory in the file manager, and vice versa, using menu commands and keyboard shortcuts:

Keyboard shortcut	Function	Use this shortcut to...
Ctrl+C	Copy	Copy selected text to the Clipboard.

- Move (drag) or copy (Ctrl +drag) a file or directory from the file manager to a directory in the Project tool window.
- Move (drag) or copy (Ctrl +drag) a file from the Project tool window to a directory in the file manager.
- Open any file for editing, by dragging it from a file manager to the editor.

Language and Framework - Specific Guidelines

This part part provides descriptions of the actions required to fulfil certain tasks using the frameworks integrated into PhpStorm:

- PHP-Specific Guidelines
- CoffeeScript Support
- Copyright
- Dart Support
- Databases and SQL
- Docker
- IntelliLang
- JavaScript-Specific Guidelines
- Markup Languages and Style Sheets
- Node.js
- Testing RESTful Web Services
- TypeScript Support
- TextMate
- Using Ext JS Framework
- Using the Flow Type Checker
- Using Bower Package Manager
- Using ReactJS in JavaScript and TypeScript
- Vagrant: Working with Reproducible Development Environments
- XPath and XSLT Support

PHP-Specific Guidelines

This part describes some procedures that are specific for developing PHP applications and some preliminary steps that are required to configure PHP development environment.

To develop an application using PHP

Follow these general steps:

1. Configure the PHP development environment .
2. Enable PHP support in PhpStorm.
3. Set up a project. You can do it in two ways:
   - Create a project from scratch .

     Tip To run and debug your application on a local Web server, create the project root below the Web server document root. Thus your application sources will be "visible" for the local Web server.
   - Download the sources and arrange them in a project .
4. Create and configure the required data sources (see Managing Data Sources).
5. Populate the application using provided coding assistance.
6. Deploy the application.

   Tip With PhpStorm, you can flexibly configure deployment of PHP applications. For example, you can set up your PHP project on a local Web server from the very beginning, or develop and test an application locally and then upload it to a remote Web server, etc.
7. Run the application. You can do it in several ways:
   - From PhpStorm using a run configuration of the type PHP Web Application to view application output in a browser.
   - From PhpStorm using a PHP Console run configuration to view the application output in the Run tool window.
   - From PhpStorm, using a built-in Web server . This approach saves your time and effort because you do not need to deploy the application sources.

The following optional steps may be helpful:

- Set up unit testing in your project.
- Install and configure a debugging engine and specify the debugger options .
- Debug the application.
- Enable and use integration with PHP-specific command line tools .

Configuring PHP Development Environment

In this section:

- Configuring PHP Development Environment
  - Prerequisites
  - Configuring PHP development environment
- Installing an AMP Package
- Installing Components Separately

Prerequisites

Warning! This topic gives general guidelines in configuring environment for developing and testing PHP applications locally. These instructions by no means apply to configuring production environment, which is outside the scope of this Help.

PHP development requires the following software installed and configured:

- A Web server and a PHP engine are mandatory.
  Starting with version 5.4, PHP interpreters contain a built-in Web server. The server is by no means intended for production but for development and testing purposes only.
- A database server, if your application will use a database.
- A debugging tool, if you are going to debug your application.
- A command line tool, if you are going to run PHP commands from the command line.

Configuring PHP development environment

To set up the PHP development environment, follow these general steps:

1. Download, install, and configure the Web server, the PHP engine, and the MySQL server. Do one of the following:
   - Download, install, and configure the desired AMP package (Apache, MySQL, PHP).
   - Install and configure each component separately , then check your installation .

2. Optionally, perform these steps:
   - Install and configure a debugging engine .
   - Install or create and enable a PHP-specific command line tool .
   - Install and enable the PHPUnit tool .

Installing an AMP Package

AMP packages are operating system-specific. The most common ones are:

- XAMPP for Windows.

  Tip It is recommended that you use version 1.7.1 or later.

- The LAMP package compatible with the Linux distribution used.
- MAMP for OS X.

The installation procedure may differ depending on the operating system used, follow the installation instructions provided.

Warning! If you are using Windows Vista, avoid installing the package in the `Program Files` folder. This folder is write-protected by default, which means that no files can be placed on the server and further processed by the PHP engine.

To install and configure an AMP package

1. Download and install the desired AMP package.

2. Use the AMP control pane to start the components.
   If the server does not start, most likely a port conflict takes place. By default, the Apache HTTP server listens to port 80. This port can be already used by other services, for example, Skype. To solve the issue, update the server configuration file as follows:
   - Locate the line `Listen 80` and change it to, e.g., `Listen 8080`.
   - Locate the line `ServerName localhost:80` and change it accordingly, in this example to `ServerName localhost:8080`.

   Save the configuration file and restart the Web server.

3. To check your installation, open your browser and type the following URL address: `http://localhost:<port number>`. The AMP welcome page appears.

Installing Components Separately

The installation procedure for PHP environment components may differ depending on the operating system used, follow the installation instructions provided.

In this section:

- Installing and configuring the components separately
- Checking the Web server installation
- Checking the PHP engine installation

To install and configure each component separately

1. Download and install a Web server . The most common choice is the Apache HTTP server .

2. Download and install the PHP engine . During the installation, specify the Web server you are going to use and its home directory.

   Tip To enable the use of the MySQL database server, select the Complete installation option or select the MySQL and MySQLi items in the Extensions list on the corresponding page of the installation wizard..

3. Download, install, and configure a database server .

To check the Web server installation

- To make sure that the Web server has been installed correctly, start the server, open your browser, and type the following URL address: `http://localhost`
  The Test page for Apache installation should appear.

  Tip If the server does not start, most likely a port conflict takes place. Resolve the conflict as during an AMP package installation.
  - If you have changed the default port 80, specify the port number explicitly: `http://localhost:<port number>`

To check the PHP engine installation

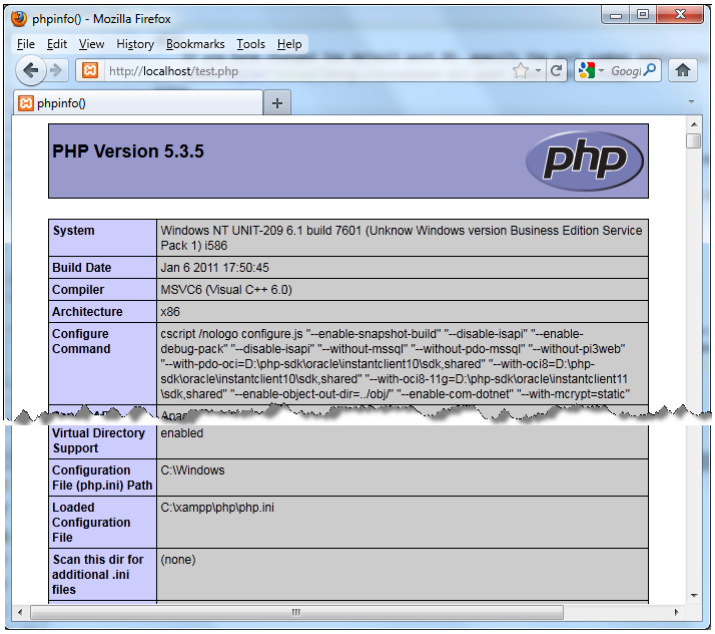
Finally, make sure that the PHP engine has been installed successfully and interacts with the Web server correctly.

1. Using a text processor of your choice, create a file `test.php` and type the following text:

```
<?php
echo phpinfo();
?>
```

2. Save the file on the Web server, in the folder the PHP engine looks at.

3. Run the browser and enter the following URL address: `http://localhost:<port number>/test.php` . The page that opens should show detailed information on the version of PHP engine used, the location of the configuration files, etc.



Enabling PHP support

In this section:

- Enabling PHP Support
  - Introduction
  - Before you start
- Configuring Remote PHP Interpreters
- Configuring Local PHP Interpreters
- Enabling PHP Support in a Project

Introduction

After you have downloaded, installed, and configured the tools that make your development environment, you need to integrate them with PhpStorm and activate their support in the project you are going to work with. Basically, all you need to start is configure your PHP installation as an interpreter. You can have several PHP engines installed on your computer, or on remote hosts, or in Vagrant virtual development environments and configured as interpreters in PhpStorm, see Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters . However, within a project, you can have only one of these interpreters activated.

Optionally, you can also enable support of other PHP development tools, see Configuring Synchronization with a Web Server , Configuring a Debugging Engine , PHP-Specific Command Line Tools , Testing PHP Applications .

Before you start

- Make sure the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
- Synchronization with Web servers is provided via the Remote Hosts Access bundled plugin. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Configuring Remote PHP Interpreters

On this page:

- Prerequisite
- Important notes
- Configuring a remote PHP interpreter on a host accessible through SSH connection
  - Before you start
  - Configuring a PHP interpreter using SSH credentials
- Configuring a remote PHP interpreter in a Vagrant environment instance
  - Before you start
  - Configuring a PHP interpreter in a Vagrant instance
- Configuring a remote PHP interpreter on a remote host accessible through SFTP
  - Before you start
  - Configuring a remote PHP interpreter based on an SFTP server access configuration
- Configuring custom mappings

Prerequisite

Before you start working with remote interpreters, make sure that the SSH Remote Run plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Important notes

The term remote PHP interpreter denotes a PHP engine installed on a remote host or in a virtual environment set up in a Vagrant instance. The term remote PHP interpreter is used opposite to local PHP interpreters that are installed on your computer, see Configuring Local PHP Interpreters .

You can access a remote PHP interpreter in the following ways:

- Using SSH credentials to access the host where the PHP interpreter is installed.
- Through access to the corresponding Vagrant instance.
- According to a Server Access Configuration . This approach is also helpful if you are going to synchronize your project sources with the Web server on the target remote host.

Configuring a remote PHP interpreter on a host accessible through SSH connection

Before you start

1. Configure access to an `ssh` server on the target remote host and make sure this server is running.
2. Make sure the PHP Remote Interpreter plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Configuring a PHP interpreter using SSH credentials

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click the ... button next to the Interpreter drop-down list in the Development environment section.
3. In the Interpreters dialog box that opens, click the Add toolbar button + in the left-hand pane, then choose Remote on the context menu.
4. In the Configure Remote PHP Interpreter dialog box that opens, choose the SSH Credentials method.
5. Specify the name of the remote host and the port which the SSH server listens to. The default port number is 22.
6. Specify your credentials to access the remote host in accordance with the credentials received during the registration on the server. Type your user name and choose the authentication method:
   - To access the host through a password, choose Password from the Auth type drop-down list and type the password.
   - To access the host through a pair of SSH keys, choose Key pair, then specify the path to the file where your private key is stored or the passphrase if you have configured it during the generation of the key pair.

   To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm first attempts to retrieve path mappings itself by processing all the available application-level configurations. If PhpStorm finds the configurations with the same host as the one specified above, in the Host field, the mappings from these configurations are merged automatically. If no configurations with this host are found, PhpStorm displays an error message informing you that path mappings are note configured.

   To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the server access configuration in question, switch to the Mappings tab, and map local folders to folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.
7. Specify the location of the PHP executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/php` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the PHP home directory must be open for edit.
8. When you click OK, PhpStorm checks whether the PHP executable is actually stored in the specified folder.
   - If no PHP executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the PHP executable is found, you return to the Interpreters where the installation folder and the detected version of the PHP interpreter are displayed.
9. Optionally, customize the configuration settings of the installation in the Additional area. In the Debugger extension text box, specify the path to Xdebug. This enables PhpStorm to activate Xdebug when it is necessary is you have disabled it in the `php.ini` file, see Configuring Xdebug for Using in the On-Demand Mode .
   - In the Configuration options field, compose a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. The directives specified in this field override the default directives generated by PhpStorm, such as `-dxdebug.remote_enable=1` , `-dxdebug.remote_host=127.0.0.1` , `-dxdebug.remote_port=9001` , `-dxdebug.remote_mode=req` .
     For example, if you specify the `-dxdebug.remote_mode=jit` directive it will override the default `-dxdebug.remote_mode=req` directive and thus switch Xdebug to the Just-In-Time (JIT) mode, see Debugging in the Just-in-Time Mode for details.

To do that, click the Browse button ... next to the Configuration options field, and then create a list of entries in the Configuration Directives dialog box, that opens.

- To add a new entry, click the Add button +. In the new line, that is added to the list, specify the name of the new entry and its value in the Name and Value text boxes respectively.
  You can add as many entries as you need, just keep in mind that they will be transformed into a command line with its length limited to 256 characters.
- To delete an entry, select it in the list and click the Remove button —.
- To change the order of entries, use the Up and Down buttons.

Upon clicking OK, you return to the Interpreters dialog box, where the entries are transformed into a command line.

Configuring a remote PHP interpreter in a Vagrant environment instance

Before you start

1. Make sure that Vagrant and Oracle's VirtualBox are downloaded, installed, and configured on your computer as described in Vagrant: Working with Reproducible Development Environments .
2. Make sure the Vagrant plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Make sure that the parent folders of the following executable files are added to the system PATH variable:
   - `vagrant.bat` or `vagrant` from your Vagrant installation. This should be done automatically by the Vagrant installer.
   - `VBoxManage.exe` or `VBoxManage` from your Oracle's VirtualBox installation.
4. Configure the PHP development environment in the Vagrant instance to be used.

Learn more about using Vagrant with PhpStorm in Vagrant: Working with Reproducible Development Environments .

Configuring a PHP interpreter in a Vagrant instance

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click the ... button next to the Interpreter drop-down list in the Development environment section.
3. In the Interpreters dialog box that opens, click the Add toolbar button + in the left-hand pane, then choose Remote on the context menu.
4. In the Configure Remote PHP Interpreter dialog box that opens, choose the Vagrant method.
5. Specify the Vagrant instance folder which points at the environment you are going to use. Technically, it is the folder where the VagrantFile configuration file for the desired environment is located. Based on this setting, PhpStorm detects the Vagrant host and shows it as a link in the Vagrant Host URL read-only field.
   To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm evaluates path mappings from the `VagrantFile` configuration file.
6. Specify the location of the PHP executable file in accordance with the development environment configured in the selected Vagrant instance. By default PhpStorm suggests the `/usr/bin/php` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the PHP home directory must be open for edit.
7. When you click OK, PhpStorm checks whether the PHP executable is actually stored in the specified folder.
   - If no PHP executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the PHP executable is found, you return to the Interpreters where the installation folder and the detected version of the PHP interpreter are displayed.
8. Optionally, customize the configuration settings of the installation in the Additional area. In the Debugger extension text box, specify the path to Xdebug. This enables PhpStorm to activate Xdebug when it is necessary is you have disabled it in the `php.ini` file, see Configuring Xdebug for Using in the On-Demand Mode .
   - In the Configuration options field, compose a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. The directives specified in this field override the default directives generated by PhpStorm, such as `-dxdebug.remote_enable=1` , `-dxdebug.remote_host=127.0.0.1` , `-dxdebug.remote_port=9001` , `-dxdebug.remote_mode=req` .
     For example, if you specify the `-dxdebug.remote_mode=jit` directive it will override the default `-dxdebug.remote_mode=req` directive and thus switch Xdebug to the Just-In-Time (JIT) mode, see Debugging in the Just-in-Time Mode for details.

     To do that, click the Browse button ... next to the Configuration options field, and then create a list of entries in the Configuration Directives dialog box, that opens.
     - To add a new entry, click the Add button +. In the new line, that is added to the list, specify the name of the new entry and its value in the Name and Value text boxes respectively.
       You can add as many entries as you need, just keep in mind that they will be transformed into a command line with its length limited to 256 characters.
     - To delete an entry, select it in the list and click the Remove button —.
     - To change the order of entries, use the Up and Down buttons.

     Upon clicking OK, you return to the Interpreters dialog box, where the entries are transformed into a command line.

Configuring a remote PHP interpreter on a remote host accessible through SFTP

Before you start

1. Make sure a `ssh` server is running on the target remote host and you have configured access to it.
2. Make sure the PHP Remote Interpreter and Remote Hosts Access plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Make sure you have at least one PhpStorm-wide server access configuration of the type SFTP to establish access to the target host. To make a configuration available in all PhpStorm projects, clear the Visible only for this project check box in the Deployment: Connection Tab . See Creating a Remote Server Configuration for details.

Configuring a remote PHP interpreter based on an SFTP server access configuration

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click the ... button next to the Interpreter drop-down list in the Development environment section.
3. In the Interpreters dialog box that opens, click the Add toolbar button + in the left-hand pane, then choose Remote on the context menu.

4. In the Configure Remote PHP Interpreter dialog box that opens, choose the Deployment Configuration method.
5. From the Deployment Configuration drop-down list, choose the server access configuration of the type SFTP according to which you want PhpStorm to connect to the target host. If the settings specified in the chosen configuration ensure successful connection, PhpStorm displays the URL address of the target host as a link in the Deployment Host URL field.
   To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . By default, PhpStorm retrieves path mappings from the chosen server access (deployment) configuration. If the configuration does not contain path mappings, PhpStorm displays the corresponding error message.

   To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the relevant server access configuration, switch to the Mappings tab, and map the local folders to the folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.
6. Specify the location of the PHP executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/php` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the PHP home directory must be open for edit.
7. When you click OK, PhpStorm checks whether the PHP executable is actually stored in the specified folder.
   - If no PHP executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the PHP executable is found, you return to the Interpreters where the installation folder and the detected version of the PHP interpreter are displayed.
8. Optionally, customize the configuration settings of the installation in the Additional area. In the Debugger extension text box, specify the path to Xdebug. This enables PhpStorm to activate Xdebug when it is necessary is you have disabled it in the `php.ini` file, see Configuring Xdebug for Using in the On-Demand Mode .
   - In the Configuration options field, compose a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. The directives specified in this field override the default directives generated by PhpStorm, such as `-dxdebug.remote_enable=1` , `-dxdebug.remote_host=127.0.0.1` , `-dxdebug.remote_port=9001` , `-dxdebug.remote_mode=req` .
     For example, if you specify the `-dxdebug.remote_mode=jit` directive it will override the default `-dxdebug.remote_mode=req` directive and thus switch Xdebug to the Just-In-Time (JIT) mode, see Debugging in the Just-in-Time Mode for details.

     To do that, click the Browse button ... next to the Configuration options field, and then create a list of entries in the Configuration Directives dialog box, that opens.
     - To add a new entry, click the Add button +. In the new line, that is added to the list, specify the name of the new entry and its value in the Name and Value text boxes respectively.
       You can add as many entries as you need, just keep in mind that they will be transformed into a command line with its length limited to 256 characters.
     - To delete an entry, select it in the list and click the Remove button —.
     - To change the order of entries, use the Up and Down buttons.

     Upon clicking OK, you return to the Interpreters dialog box, where the entries are transformed into a command line.
9. Click the Show phpinfo button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed.

Configuring custom mappings

If you use an interpreter accessible through SFTP connection or located on a Vagrant instance, the mappings are automatically retrieved from the corresponding deployment configuration or `Vagrantfile` . To specify additional mappings:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. From the Interpreter drop-down list, choose the remote interpreter for which you want to customize the mappings. The Path Mappings read-only field shows the path mappings retrieved from the corresponding deployment configuration or `Vagrantfile` , see the chapters above. To specify the custom mappings, click ... next to the Path Mappings field.
3. The Edit Project Path Mappings Dialog that opens, shows the path mappings retrieved from the deployment configuration or `Vagrantfile` . These mappings are read-only.
   - To add a custom mapping, click + and specify the path in the project and the corresponding path on the remote runtime environment in the Local Path and Remote Path fields respectively. Type the paths manually or click ... and select the relevant files or folders in the dialog box that opens.
   - To remove a custom mapping, select it in the list and click —.

Configuring Local PHP Interpreters

The term local PHP interpreter denotes a PHP engine installed on your computer opposite to remote PHP interpreters that can be installed on a remote host or in a virtual environment set up in a Vagrant instance, see Configuring Remote PHP Interpreters .

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click the ... button next to the Interpreter drop-down list in the Development environment section.
3. In the Interpreters dialog box that opens, click the Add toolbar button + in the left-hand pane, then choose Local on the context menu. If you already have a local interpreter configured in PhpStorm, it is also shown on the menu and the menu item changes to Other Local.
4. In the right-hand pane of the dialog box, specify the PHP interpreter's settings.
   1. In the Name text box, type the identifier to distinguish the interpreter from others, for example, `php_installation_<version>`.
   2. Specify the PHP engine installation directory in the PHP Home field. Type the path manually or click the Browse button ... and choose the location in the Choose PHP Home dialog box, that opens.
      PhpStorm displays the version of the PHP engine detected in the specified folder and the debugger associated with this PHP engine in the `php.ini` file.
5. Optionally, customize the configuration settings of the installation in the Additional area. In the Debugger extension text box, specify the path to Xdebug. This enables PhpStorm to activate Xdebug when it is necessary is you have disabled it in the `php.ini` file, see Configuring Xdebug for Using in the On-Demand Mode .
   - In the Configuration options field, compose a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. The directives specified in this field override the default directives generated by PhpStorm, such as `-dxdebug.remote_enable=1`, `-dxdebug.remote_host=127.0.0.1`, `-dxdebug.remote_port=9001`, `-dxdebug.remote_mode=req`.
     For example, if you specify the `-dxdebug.remote_mode=jit` directive it will override the default `-dxdebug.remote_mode=req` directive and thus switch Xdebug to the Just-In-Time (JIT) mode, see Debugging in the Just-in-Time Mode for details.

     To do that, click the Browse button ... next to the Configuration options field, and then create a list of entries in the Configuration Directives dialog box, that opens.
     - To add a new entry, click the Add button +. In the new line, that is added to the list, specify the name of the new entry and its value in the Name and Value text boxes respectively.
       You can add as many entries as you need, just keep in mind that they will be transformed into a command line with its length limited to 256 characters.
     - To delete an entry, select it in the list and click the Remove button —.
     - To change the order of entries, use the Up ↑ and Down ↓ buttons.

     Upon clicking OK, you return to the Interpreters dialog box, where the entries are transformed into a command line.
6. Click the Show phpinfo button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed.

Enabling PHP Support in a Project

On this page:

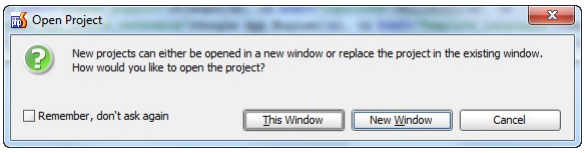
- Creating an empty PHP project
- Creating a PHP project from existing sources
- Choosing the PHP interpreter and the language level manually

Creating an empty PHP project

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose PHP Empty Project.
3. In the right-hand pane:
   - In the Location text box, specify the path to the project folder where the project-related files will be stored.
   - In the PHP Language Level drop-down list, specify the PHP functionality scope to get coding assistance for. Each functionality scope is associated with PHP version that supports this functionality. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported.
     No correlation between the PHP version used in the project and the language level is enforced. Although the language version of each interpreter is detected automatically, you can still tell PhpStorm to provide you with coding assistance that corresponds to another language level. However, if you attempt to use a code construct that is not supported by the specified language level, PhpStorm suggests a Switch to PHP <version> quick-fix .
   - From the Interpreter drop-down list, choose the PHP interpreter to use in the current project. The list contains all the currently configured local and remote PHP interpreters. See Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters for details.
   - The Include Path area shows a list of paths to PHP-related items below the PHP home directory. The specified include paths will be used:
     - By the `require()`, `include()`, `fopen()`, `file()`, `readfile()`, and `file_get_contents()` functions when looking for files to use.
     - By PhpStorm when resolving references to included files.

     Use the Add + and Remove — buttons to manage the contents of the list. Use the Up and Down buttons to change the order of items in the list.

   When you click Create, PhpStorm generates a project stub and opens it either in the current window or in a new one depending on your choice in the information dialog box that opens:



   Refer to the section Opening Multiple Projects for details.

See also Configuring PHP Namespaces in a Project

Creating a PHP project from existing sources

Besides creating an empty PHP project, you can re-use your previously developed sources.

- When the files to re-use are already on your machine, all you need is just open the folder where they are stored:
  1. On the main menu, choose File | Open.
  2. In the dialog box that opens, select the folder where the required sources are stored.

  See Creating a Project from Existing Local Sources for details.
- When the files to re-use are on a remote host:
  1. Configure synchronization with this remote host, see Configuring Synchronization with a Web Server .
  2. Download the required sources and arrange them in a PhpStorm project on the fly, see Creating a Project from Downloaded Files .

Choosing the PHP interpreter and the language level manually

To start PHP development in a project, you need to appoint one of the previously configured local or remote PHP interpreters to use and choose the PHP language level which determines the scope of PHP functionality to get coding assistance for. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported.

To use PHP-related items from outside your project content root, configure include paths as described in Configuring Include Paths .

If you created your project through the New Project dialog, you have already specified the language level and possibly the interpreter and the include paths. However, if your project was created around some existing sources, you need to specify these settings manually.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks. Open the PHP page: choose File | Settings and in the Settings dialog box that opens, click PHP under Languages & Frameworks.
2. In the PHP page that opens, choose the PHP installation to use from the Interpreter drop-down list.
   - To make sure that the configuration you have chosen points at the relevant installation, click the Reload button next to the drop-down list. If no PHP engine is actually installed at the specified directory, PhpStorm displays the corresponding error message.
   - Click the Show phpinfo button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed.
3. In the PHP language level drop-down list, specify the PHP functionality scope to get coding assistance for. Each functionality scope is associated with PHP version that supports this functionality. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported.
   No correlation between the PHP version used in the project and the language level is enforced. Although the language version of each interpreter is detected automatically, you can still tell PhpStorm to provide you with coding assistance that corresponds to another language level. However, if you attempt to use a code construct that is not supported by the specified language level, PhpStorm suggests a Switch to PHP <version> quick-fix .

Using Distributed Configuration Files (.htaccess)

On this page:

- Introduction
- Associating a name pattern with the distributed configuration file type
- Important notes

Introduction

PhpStorm provides syntax highlighting, formatting, code completion and documentation lookup while editing distributed configuration files .

Distributed configuration files are used to make directory-based changes to the HTTP Apache Server configuration and usually have the name `.htaccess`.

If you are using a file with the name `.htaccess`, PhpStorm recognizes it as distributed configuration file and provides full coding assistance for it, so no additional steps are required from your side.

If you want to use a file with another name, you need to associate this full name or the corresponding pattern with the distributed configuration file type . After that, PhpStorm will treat any file with the name matching the specified pattern as a distributed configuration file and process it accordingly.

Associating a name pattern with the distributed configuration file type

To associate a name pattern with the distributed configuration file type

1. Open the Settings/Preferences and click File Types.
2. In the File Types page that opens, select Apache config files from the Recognized File Types list.
3. In the Registered Patterns area, click +.
4. In the Add wildcard dialog box that opens, specify the pattern that defines the extensions of your distributed configuration files.
5. Click OK. PhpStorm returns you to the File Types page where the specified pattern is added to the Registered Patterns list.

Important notes

Please note the following:

- By default, the Registered Patterns list contains one item `htaccess`.
- To discard a pattern, select it in the list and click the Remove button.
- To change a pattern, select it in the list, click the Edit button, and update the pattern as necessary in the Add wildcard dialog box that opens.

Configuring Include Paths

Include paths are used for code completion and reference resolution in some functions/methods that use file paths as arguments (for example, `require()` or `include()`). You also need to configure PEAR as an include path to run the PHPUnit test framework from it when the Load from include path option is chosen on the PHPUnit page of the Settings / Preferences Dialog .

To configure include paths:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. In the PHP page that opens, click the Update include paths button.
3. In the Include paths area, that becomes available upon pressing the Update include paths button, configure the list of include paths.
   - To add a path to the list, click the Add button and select the required directory in the dialog box, that opens.
   - To remove a path from the list, select the required item and click the Remove button.

Configuring PHP Namespaces in a Project

On this page:

– Introduction

– Detecting namespace roots automatically

– Configuring namespace roots manually

Introduction

PhpStorm comes with a configurable convention which specifies that the project root folder is also the root for the packages and namespaces. In other words: the project root folder is by default marked as Source and every directory created under it is considered a separate namespace. This complies with the PSR-0 , also known as the Autoloading Standard, which prescribes that classes and namespaces in PHP should match the directory and file structure, and vice-versa.

According to PSR-4 , any directory can be explicitly assigned a namespace prefix. With this project structure, autoloaders in different PHP frameworks become interoperable.

Detecting namespace roots automatically

When you open a project that contains at least one file with a namespace, PhpStorm displays a message with a proposition to set the namespace root.

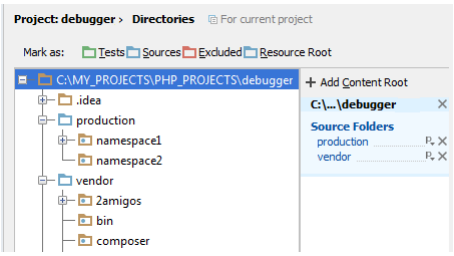
Accordingly, when no namespace root has been configured yet and you create a class, PhpStorm proposes to configure the namespace root.

You can also trigger namespace root detection by choosing Code | Detect PSR-0 Namespace Roots on the main menu. The Directories dialog box that opens, shows the folders under the project root folder with the project root folder marked as Source, which means that it is the root for all the namespaces in it. Accept the settings by clicking OK or configure the namespace root manually as described below. Learn more about marking folders in Configuring Folders Within a Content Root .

Configuring namespace roots manually

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Directories under Project:<Project Name>.
2. The right-hand pane of the Directories page that opens shows all the content roots configured in the project. As a rule, there is one content root which is the root folder of the current project.
   The central pane shows all the folders under the selected content root. Select the folder to be treated as the namespace root and click Sources.

   It is perfectly fine to configure multiple source roots: PhpStorm will treat each of those as a namespace root and provide namespace hints for files underneath them. For example every subfolder of a vendor folder can be marked as a namespace root. In the example below, we have two namespace roots: one for the actual application and one for a vendor folder:



3. To configure a namespace prefix according to PSR-4 , click next to the relevant source folder (namespace root), and specify prefix to use in the Edit Root Properties Dialog box. To have the prefix applied to the automatically generated code too, select the For generated sources check box.

PHP-Specific Coding Assistance

On this page:

– Converting long array syntax to short

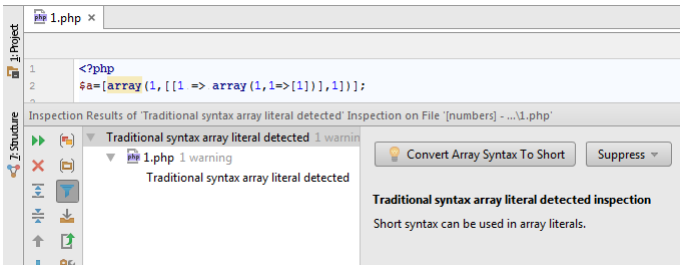
Converting long array syntax to short

Suppose you have an array declared in the traditional literal style:

```
<?php
                    $a=[array(1,[[1 => array(1,1=>[1])],1])];
```

You can detect all the traditional literal style declarations in your code and convert them into short style:

1. On the main menu, choose Code | Run Inspection by Name, or press `Ctrl+Shift+Alt+I`.
2. In the pop-up frame that opens, start typing the inspection name Traditional syntax array literal. As you type, the suggestion list shrinks to show the matching inspection only. Press `?`. The Specify Inspection Scope dialog box opens.
3. In the Inspection scope area, specify which files should be inspected.
   - To have the source code of the entire project inspected, select the Whole Project option.
   - If you are using version control integration , you can choose to only inspect uncommitted files.
   - To run an inspection for the currently opened file, or the file(s)/folder(s) selected in the Project view, select the File/Module <name> option.
   - To apply inspect code in a specific scope , select the Custom scope option, then choose the desired scope from the drop-down list or click the Browse button ... and configure a new scope in the Scopes dialog box.
4. To have test source files inspected too, select the Include test sources check box.
5. Click OK to run the inspection.
6. The Inspection Tool Window opens showing the list of files where the traditional literal style declarations have been detected:



Click Convert array syntax to short. The code is updated as follows:

```
<?php
                    $a=[ [ 1,
                    [ [ 1 => [
                    1, 1=>[ 1] ] ], 1] ] ];
```

Running PHP Applications

You can run your PHP application in several ways:

- From PhpStorm using a run configuration of the type PHP Web Application to view application output in a browser.
- From PhpStorm using a PHP Console run configuration to view the application output in the Run tool window.
- From PhpStorm, using a built-in Web server . This approach saves your time and effort because you do not need to deploy the application sources.

PhpStorm enables running entire PHP applications as well as particular classes or files .

To run an entire PHP application, perform these general steps

1. Define a PHP run configuration:
   - To launch the application via a console, create a PHP Console configuration.
   - To run the application on a local or a remote Web server, create a PHP Web Application configuration.
2. Deploy the PHP application on a local or remote server, if applicable.
3. Run the application with the required run configuration.

To run a particular PHP class or file, do one of the following

- Open the desired class or file in the editor and choose Run <file name> on the context menu or press `Ctrl+Shift+F10`.
- Select the class or file in the Project view and choose Run <file name> on the context menu of the selection.

PHP Built-In Web Server

On this page:

- Introduction
- Enabling a built-in Web server
- Creating a run configuration of the type PHP Built-in Server
- Running a PHP application on a built-in Web server

Introduction

Starting with version 5.4.0, PHP interpreters incorporate a Web server functionality. As soon as your project root folder or any other folder is declared the built-in Web server document root, all the files under this folder are submitted to the PHP interpreter for processing. All you need, is enable the built-in Web server in your project, create a run configuration of the type PHP Built-in Web server, and launch the application with this configuration.

Integration with the built-in Web server is supported PhpStorm 4.0 and higher.

Enabling a built-in Web server

1. Download PHP 5.4.0 or higher.
2. Register the downloaded PHP interpreter in PhpStorm .
3. Open your project .
4. Appoint the PHP 5.4.0 interpreter configuration for use in the project.

Creating a run configuration of the type PHP Built-in Server

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Press `Shift+Alt+F10`, then press `0` to display the Edit Configuration dialog box or select the configuration from the pop-up window and press `F4`.
2. Click + on the toolbar or press `Insert`. From the drop-down list, select the PHP Built-in Web Server configuration type. The PHP Built-in Web Server dialog box opens.
3. Specify the configuration name.
4. Specify the server connection settings:
   1. In the Host text box, type `localhost`.
   2. Use the Port spin box to specify the number of the port where the built-in web server runs. The port number should be set to 1024 or higher. By default this port is set to the default PhpStorm port through which PhpStorm accepts connections from services (`63342`).
5. Specify the folder that will be considered server document root . The folders that are under this document root in the project tree will be submitted to the PHP interpreter. The server document root folder will be accessed through HTTP at the above specified host:port.
   In the Document root text box, specify the path to the relevant folder. This can be the project root or any other folder under it.

   Optionally, customize the run configuration:
   - To have a specific script always executed upon the server start-up, select the Use router script check box and specify the path to the script in the text box.
   - To customize the behavior of the current PHP interpreter, in the Command Line area specify the options and arguments to be passed to the PHP executable file.

Running a PHP application on a built-in Web server

1. Run the application. To do that, select the built-in Web server run configuration from the Run/Debug Configuration drop-down list on the main tool bar, and then click the Run button ▶.
2. Launch the browser of your choice and open the starting page of your application by typing its URL address in the following format:

```
http://localhost:<port>/<path to the required file relative to the server document root>
```

Creating PHP Documentation Comments

On this page:

- Basics
  - Enabling generation of documentation comments
- Generating a PHPDoc block for a method or a function
- Creating tags in a PHPDoc comment block
- Inspecting PHPDoc comments
- Configuring formatting inside PHPDoc comments

Basics

PhpStorm creates stubs of PHPDoc blocks on typing the opening tag `/**` and pressing `Enter` or pressing `Alt+Insert` and appointing the method.function to document.

If this feature is applied to a method or a function , `@param` , `@access` , `@static` , `@throws` , `@return` , `@var` , and `@abstract` tags are created. In any other places PhpStorm adds an empty documentation stub.

If you need additional PHP-specific tags , PhpStorm provides code completion that suggests tag names that are relevant in the current context. If a certain tag has multiple values, the same code completion provides a list of available values.

In PHPDoc comments, PhpStorm supports formatting options in compliance with ZEND, PEAR, and other standards.

PHPDoc comments in your source code are available for the Quick Documentation Lookup feature and open for review on pressing `Ctrl+Q`.

To enable or disable generation of documentation comments

1. Open the Settings dialog box , expand the Editor node, then expand General node, and click the Smart Keys page.
2. In the Enter section, select or clear Insert documentation comment stub check box.

Generating a PHPDoc block for a method or a function

1. To invoke PHPDoc block generation, do one of the following:
   - Place the caret before the method or function declaration, type the opening block comment `/**` , and press `Enter` .
   - On the context menu anywhere in the editor, choose Generate, then choose Generate PHPDoc blocks, and choose the method or function to generate comments for.
   - Press `Alt+Insert` , then choose Generate PHPDoc blocks, and choose the method or function to generate comments for.

   PhpStorm analyzes the appointed method or function, where possible extracts the data for method parameters, return values, variables, etc., and on this basis generates a stub of a documentation block.
2. Describe the listed parameters and return values, where necessary.

   Tip PhpStorm checks syntax in the comments and treats it according to the PHP Inspections settings.

Creating tags in a PHPDoc comment block

PhpStorm analyzes the appointed method or function, where possible extracts the data for method parameters, return values, variables, etc., and on this basis generates a stub of a documentation block. If necessary, you can fill in the missing information.

1. In a PHPDoc block, select the desired empty line and press `Ctrl+Space` .
2. Select the relevant tag from the suggestion list.
3. If the entered tag has several values, press `Ctrl+Space` and select the desired value from the suggestion list.

Inspecting PHPDoc comments

PhpStorm provides a set of predefined code inspections targeted at PHPDoc blocks. These inspections check whether classes, methods, functions, variables, and constants are supplied with a PHPDoc comment and whether the tags in the comment match the documented item.

- To enable or disable an inspection:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Inspections under Editor.
  2. On the Inspections page that opens, expand the PHPDoc node under the PHP node. The list of predefined inspections is displayed.
  3. To enable or disable an inspection, select or clear the check box next to it.
- To have PhpStorm check that PHPDoc comments are provided for all code constructs of a certain type:
  1. Select the check box next to the Missing PHPDoc Comment inspection.
  2. In the Options area, select the check boxes next to the required code construct type: class, method, function, variable, or constant.
  3. To suppress reporting a Missing PHPDoc Comment error if a method or function does not contain any parameters and/or return values, select the Skip if empty.

Configuring formatting inside PHPDoc comments

You can configure the appearance of PHPDoc comments and the presentation of class names in the Settings dialog box, on the PHPDoc tab of the Code Style. PHP under the Code Style node. Note that the tag for properties is no longer configurable, the default @var tag is inserted automatically. See https://github.com/phpDocumentor/fig-standards/pull/55 for details.

- Open the Settings dialog box , click Code Style, then click PHP, and switch to the PHPDoc tab.

1. Open the Settings dialog box , click Code Style, then click PHP , and switch to the PHP Doc tab.
2. Configure the alignment by selecting or clearing the check boxes in the tab.
3. Specify how you want PhpStorm to present class names for properties, function parameters, return and throws values, etc. by selecting or clearing the Use fully-qualified class names check box.

Viewing PHPDoc Comments

Quick Documentation Lookup helps you get quick information for any symbol, provided that this symbol has been supplied with Documentation comments in the applicable format. PhpStorm recognizes inline documentation created in accordance with the PHP Documentation format.

In this section:

- Viewing quick documentation .
- Changing font size in the quick documentation window .
- Toolbar of the quick documentation window .

To view documentation for a symbol at caret, do one of the following

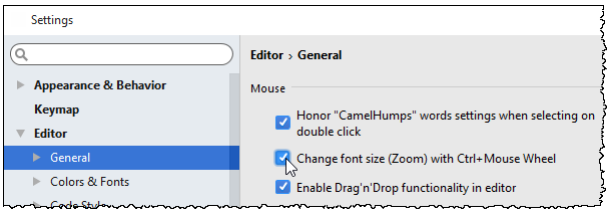
- On the main menu, choose View | Quick Documentation Lookup.
- Press `Ctrl+Q`.
- Provided that the check box Show quick doc on mouse move in the editor settings is selected, just move your mouse pointer over the desired symbol.

When you explicitly invoke code completion, then quick documentation for an entry selected in the suggestion list can be displayed automatically. The behavior of quick documentation lookup is configured in the Code Completion page of the Settings/Preferences dialog.

Tip Sequential pressing `Ctrl+Q` toggles the focus of the Quick Documentation pop-up window and Documentation Tool Window .

To change the font size of quick documentation, do one of the following

- Click in the upper-right corner of the quick documentation window, and move the slider.
- Rotate the mouse wheel while keeping the `Ctrl` key pressed.
  Note that for this feature to work, you have to enable it in the General page of the editor settings.



Refer to the section Zooming in the Editor for details.

Quick Documentation Lookup Window

Toolbar of the quick documentation lookup

The Quick Documentation Lookup window helps navigate to the related symbols via hyperlinks, and provides a toolbar for moving back and forth through the already navigated pages, changing font size, and viewing documentation in an external browser.

When pinned, the Quick Documentation Lookup turns into Documentation Tool Window , with the corresponding sidebar icon, and more controls.

Icon	Shortcut	Description
	`Left` or `Right`	Switch to the previous or next documentation page (e.g. after using hyperlinks). **Note** On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.
	`Shift+F1`	View external documentation in the default browser.
	`F4`	Switch to the item (e.g. source) that corresponds to the documentation page currently shown.
		Turn the Auto-update from source option on or off. When the option is on, the information in the tool window is synchronized with your navigation in the editor and other places in the UI.
		Click this icon to show font size slider. Move the slider to increase or decrease the font size in the quick documentation window as required.

Using PHP Code Sniffer Tool

On this page:

- Introduction
- Before you start
- Choosing the Code Sniffer script to use
- Configuring a local Code Sniffer script
- Configuring a Code Sniffer associated with a PHP interpreter
- Configuring advanced PHP Code Sniffer options
- Configuring PHP Code Sniffer as a PhpStorm inspection
- Choosing a custom coding style to check your code against
- Sharing a custom coding style with the team
- Running Code Sniffer in the batch mode
- Excluding files from Code Sniffer inspection

Introduction

In addition to built-in coding assistance, PhpStorm provides code style check through integration with the PHP Code Sniffer tool, which validates your code for consistence with a coding standard of your choice. You can appoint one of the predefined coding standards or use your own previously defined coding standard with the root directory outside the default PHP Code Sniffer's Standards directory. Moreover, you can share your custom coding style with your team fellows.

To use PHP Code Sniffer right from PhpStorm instead of from a command line, you need to register it in PhpStorm and configure it as a PhpStorm code inspection . Once installed and enabled in PhpStorm, the tool is available in any opened PHP file, so no steps are required from your side to launch it. The on-the-fly code check is activated upon every update in the file thus making it extremely easy to get rid of problems reported by PHP Code Sniffer.

Errors and warnings reported by PHP Code Sniffer on-the-fly are displayed as pop-up messages. When the tool is run in the batch mode, the errors and warnings are displayed in the Inspection tool window. Anyway, each message has the `phpcs` prefix to distinguish it from PhpStorm internal inspections.

Before you start

Make sure the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Choosing the Code Sniffer script to use

You can use PHP Code Sniffer local scripts or scripts associated with PHP interpreters. There can be a number of local and remote PHP interpreters, the one specified on the PHP page of the Settings dialog box is considered Project Default. Learn more about configuring PHP interpreters in Configuring Remote PHP Interpreters or in Configuring Local PHP Interpreters .

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Code Sniffer under PHP.
2. From the Configuration drop-down list in the Development Environment area, choose the script to use:
   - To use the script associated with a specific remote PHP interpreter, choose the name of this interpreter.
   - To use the script associated with the default project interpreter, that is, the one chosen on the PHP page of the Settings dialog box, choose By default project interpreter.
   - To use a local script, choose Local. In this case the local Code Sniffer will be executed no matter which PHP interpreter - local or remote - is used in the project. Note that there can be only one Local configuration for Code Sniffer because PhpStorm runs a script (`phpcs.bat` for Windows or `phpcs` for Linux) which contains a path to a PHP engine.

Configuring a local Code Sniffer script

1. Make sure the PEAR package manager is installed on your machine.
2. Download and install the PHP Code Sniffer . To check it, switch to the directory with the `pear.bat` file and run the following command: `phpcs --version`
   If the tool is available, you will get a message in the following format: `PHP_CodeSniffer version <version> (stable) by Squiz Pty Ltd. (http://www.squiz.net)`

   To have code checked against your own custom coding standard, create it . Store the rules and the `ruleset.xml` file that points to them in the coding standard root directory.
3. Register the local PHP Code Sniffer script in PhpStorm:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Code Sniffer under PHP.
   2. On the Code Sniffer page that opens, click ... next to the Configuration drop-down list.
   3. In the Code Sniffer dialog box that opens, specify the location of the Code Sniffer executable `phpcs.bat` or `phpcs` in the PHP Code Sniffer path text box. Type the path manually or click the Browse button ... and select the path in the dialog box, that opens.
      To check that the specified path to `phpcs.bat` or `phpcs` ensures interaction between PhpStorm and Code Sniffer, that is, the tool can be launched from PhpStorm and PhpStorm will receive problem reports from it, click the Validate button. This validation is equal to running the `phpcs --version` command. If validation passes successfully, PhpStorm displays the information on the detected Code Sniffer version.

Configuring a Code Sniffer associated with a PHP interpreter

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Code Sniffer under PHP.
2. On the Code Sniffer page that opens, click ... next to the Configuration drop-down list. The Code Sniffer dialog box opens showing a list of all the configured Code Sniffer scripts in the left-hand pane, one of them is of the type Local and others are named after the PHP interpreters with which the scripts are associated. Click + on the toolbar.
3. In the Code Sniffer by Remote Interpreter dialog box that opens, choose the remote PHP interpreter to use the associated script from. If the list does not contain a relevant interpreter, click ... and configure a remote interpreter in the Interpreters dialog box as described in Configuring Remote PHP Interpreters .
   When you click OK, PhpStorm brings you back to the Code Sniffer dialog box where the new Code Sniffer configuration is added to the list and the right-hand pane

shows the chosen remote PHP interpreter, the path to the Code Sniffer associated with it, and the advanced PHP Code Sniffer options.

Configuring advanced PHP Code Sniffer options

PhpStorm provides the ability to specify advanced PHP Code Sniffer options and thus fine tune the PHP Code Sniffer process behavior depending on the configuration of your computer and the rule sets used.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Code Sniffer under PHP.
2. In the Maximum number of messages per file text box, set the upper limit for the total number of messages to be reported for a file. All the messages above this limit will be rejected and PhpStorm will display the following warning right in the code:

```
Too many PHP Code Sniffer messages
```

3. In the Tool process timeout text box, specify how long you want PhpStorm to wait for a result from PHP Code Sniffer, whereupon the process is killed to prevent excessive CPU and memory usage.

Configuring PHP Code Sniffer as a PhpStorm inspection

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Inspections under Editor.
2. On the Inspections page that opens, select the PHP Code Sniffer validation check box under the PHP node.
3. On the right-hand pane of the page, configure the PHP Code Sniffer tool using the controls in the Options area:
   1. From the Severity drop-down list, choose the severity degree for the Code Sniffer inspection. The selected value determines how serious the detected discrepancies will be treated by PhpStorm and presented in the inspection results .
   2. In the Coding standard drop-down list, appoint the coding style to check your code against. The list contains all the coding standards installed inside the main `PHP_CodeSniffer` directory structure.
      Use one of the predefined coding standards or choose Custom to appoint your own standard .
   3. Optionally, select the Ignore warnings check box to have only errors reported and suppress reporting warnings. This option is equal to the `-n` command line argument.

Choosing a custom coding style to check your code against

You can have code checked against your own previously defined coding standard with the root directory outside the default PHP Code Sniffer's Standards directory. This root directory should contain the rules themselves and the `ruleset.xml` file that points to them.

1. Open the Settings dialog box , and click Inspections. The Inspections page opens. Select the PHP Code Sniffer validation check box under the PHP node.
2. From the Coding standard drop-down list, choose Custom.
3. Click the Browse button ...
4. In the Custom Coding Standard dialog box that opens, specify the path to the root directory of your own coding standard in the Root directory. Type the path manually or click the Browse button ... and choose the relevant folder in the dialog that opens .
   The selected root directory should contain the `ruleset.xml` file that points to the rules.

Sharing a custom coding style with the team

1. Put the root directory of your coding standard under the project root .
2. Configure Code Sniffer as a PhpStorm inspection .
3. Appoint your coding standard .
4. At the top of the Inspections page, select the Share Profile check box.
5. On the Version Control page of the Settings dialog box, put the root directory of your coding standard under version control .

Running Code Sniffer in the batch mode

1. To run the inspection , choose Code | Inspect code on the main menu. Specify the inspection scope and profile.
2. View the inspection results in the Inspection tool window. Errors and warnings reported by PHP Code Sniffer have the `phpcs` prefix to distinguish them from PhpStorm internal inspections.

Excluding files from Code Sniffer inspection

When waiting for Code Sniffer response exceeds the limit specified in the Tool process timeout field on Code Sniffer page, PhpStorm suggests adding the file to the ignore list. This list is shown on the Code Sniffer page in the Ignored files area. For each file, PhpStorm displays its name and location.

- To delete a file from the list and have Code Sniffer process it again, select the file and click the Remove file button .
- To remove all the files from the list, click the Clean the list button .

Using PHP Mess Detector

On this page:

- Introduction
- Before you start
- Choosing the Mess Detector script to use
- Configuring a local Mess Detector script
- Configuring a Mess Detector associated with a PHP interpreter
- Specifying advanced PHP Mess Detector options
- Configuring PHP Mess Detector as a PhpStorm inspection
- Specifying the rules to apply
- Running Mess Detector in the batch mode
- Excluding files from Mess Detector inspection

Introduction

In addition to built-in coding assistance, PhpStorm provides checking the source code through integration with the PHP Mess Detector tool, which detects potential problems related to code size, inconsistency, unused code, violation of naming conventions, poor design, etc.

You can have predefined rules applied or define your own custom set of rules .

Before you start

Make sure the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Choosing the Mess Detector script to use

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Mess Detector under PHP.
2. From the Configuration drop-down list in the Development Environment area, choose the script to use:
   - To use the script associated with a specific remote PHP interpreter, choose the name of this interpreter.
   - To use the script associated with the default project interpreter, that is, the one chosen on the PHP page of the Settings dialog box, choose By default project interpreter.
   - To use a local script, choose Local. In this case the local Mess Detector will be executed no matter which PHP interpreter - local or remote - is used in the project. Note that there can be only one Local configuration for Mess Detector because PhpStorm runs a script (`phpmd.bat` for Windows or `phpmd` for Linux) which contains a path to a PHP engine.

Configuring a local Mess Detector script

1. Download PHP Mess Detector. The easiest ways to do that is use the Composer Dependency Manager, see Using Composer Dependency Manager .
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Mess Detector under PHP.
3. On the Mess Detector page that opens, click ... next to the Configuration drop-down list.
4. In the Mess Detector dialog box that opens, specify the location of the Mess Detector executable `phpmd.bat` or `phpmd` in the PHP Mess Detector path text box. Type the path manually or click the Browse button ... and select the path in the dialog box, that opens. If you installed the tool through Composer, the default location is `<project root folder>\vendor\bin\phpmd.bat` .
   To check that the specified path to `phpmd.bat` or `phpmd` ensures interaction between PhpStorm and Mess Detector, that is, the tool can be launched from PhpStorm and PhpStorm will receive problem reports from it, click the Validate button. This validation is equal to running the `phpmd --version` command. If validation passes successfully, PhpStorm displays the information on the detected Mess Detector version.

Configuring a Mess Detector associated with a PHP interpreter

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Mess Detector under PHP.
2. On the Mess Detector page that opens, click ... next to the Configuration drop-down list. The Mess Detector dialog box opens showing a list of all the configured Mess Detector scripts in the left-hand pane, one of them is of the type Local and others are named after the PHP interpreters with which the scripts are associated. Click + on the toolbar.
3. In the Mess Detector by Remote Interpreter dialog box that opens, choose the remote PHP interpreter to use the associated script from. If the list does not contain a relevant interpreter, click ... and configure a remote interpreter in the Interpreters dialog box as described in Configuring Remote PHP Interpreters .
   When you click OK, PhpStorm brings you back to the Mess Detector dialog box where the new Mess Detector configuration is added to the list and the right-hand pane shows the chosen remote PHP interpreter, the path to the Mess Detector associated with it, and the advanced PHP Mess Detector options.

Specifying advanced PHP Mess Detector options

PhpStorm provides the ability to specify advanced PHP Mess Detector options and thus fine tune the PHP Mess Detector process behavior depending on the configuration of your computer and the rule sets used.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Mess Detector under PHP. The Mess Detector page opens.
2. In the Maximum number of messages per file text box, set the upper limit for the total number of messages to be reported for a file. All the messages above this limit will be rejected and PhpStorm will display the following warning right in the code:

```
Too many PHP Mess Detector messages
```

3. In the Tool process timeout text box, specify how long you want PhpStorm to wait for a result from PHP Mess Detector, whereupon the process is killed to prevent

excessive CPU and memory usage.

Configuring PHP Mess Detector as a PhpStorm inspection

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Inspections under Editor.
2. On the Inspections page that opens, select the PHP Mess Detector validation check box under the PHP node.
3. On the right-hand pane of the page, configure the PHP Mess Detector tool using the controls in the Options area:
   1. From the Severity drop-down list, choose the severity degree for the Mess Detector inspection. The selected value determines how serious the detected discrepancies will be treated by PhpStorm and presented in the inspection results .
   2. Appoint the rules to apply .

Specifying the rules to apply

You can have predefined rules applied or define your own custom set of rules.

1. Open the Settings dialog box , and click Inspections.
2. On the Inspections page that opens, select the PHP Mess Detector validation check box under the PHP node.
3. Do one of the following:
   - To use a predefined rules, in the Options area, select the check boxes next to the validations to be performed. For more details on predefined rules, see http://phpmd.org/rules/index.html .
   - To use a custom ruleset:
     1. Create and save one or several ruleset files. A valid ruleset file is an `.xml` file that contains the root element `<ruleset>` with the attribute `name` . For more details on custom rulesets, see http://phpmd.org/documentation/creating-a-ruleset.html .
     2. In the Custom rulesets area, click the Add Rule button + and select the relevant rule definition file in the dialog box that opens. When you click OK, a new item is added to the Custom rulesets list, where the File field shows the location of the selected ruleset file and the Name field shows the ruleset name retrieved from the attribute `name` within the `<ruleset>` tag.

Running Mess Detector in the batch mode

1. To run the inspection , choose Code | Inspect code on the main menu. Specify the inspection scope and profile.
2. View the inspection results in the Inspection tool window. Errors and warnings reported by PHP Mess Detector have the `phpmd` prefix to distinguish them from PhpStorm internal inspections.

Excluding files from Mess Detector inspection

When waiting for Mess Detector response exceeds the limit specified in the Tool process timeout field on Mess Detector page, PhpStorm suggests adding the file to the ignore list. This list is shown on the Mess Detector page in the Ignored files area. For each file, PhpStorm displays its name and location.

- To delete a file from the list and have Mess Detector process it again, select the file and click the Remove file button —.
- To remove all the files from the list, click the Clean the list button ✕.

Deploying PHP Applications

On this page:

- Introduction
- Configuring deployment
- Uploading files automatically

Introduction

Synchronization with Web servers is provided via the Remote Hosts Access bundled plugin. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

With PhpStorm, you can flexibly configure deployment of PHP applications. For example, you can set up your PHP project on a local Web server from the very beginning, or develop and test an application locally and then upload it to a remote Web server, etc.

By default, the sources are deployed according to the chosen configuration when you run your application. However, you can have PhpStorm upload files automatically every time they are changed. In this case the default server configuration is used.

Configuring deployment

1. Create a local or remote Web server configuration .
2. Define mappings between local folders, remote folders, and URL addresses.
3. Customize the upload procedure by specifying additional options.

Uploading files automatically

To have updated files uploaded automatically, do one of the following:

- Choose Tools | Deployment | Automatic Upload on the main menu.
- Open the Settings dialog box , and click Options below the Deployment node. In the Options dialog box, select the Upload changed files automatically to the default server check box.

Debugging PHP Applications

In this part:

- Configuring a Debugging Engine
- PHP Debugging Session
- Multiuser Debugging via Xdebug Proxies

Configuring a Debugging Engine

This section describes how to configure two most popular tools used in PHP debugging:

- Xdebug
- Zend Debugger

These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file. To open the active `php.ini` file in the editor:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click ... next to the Interpreter field.
3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.

In this part:

- Configuring Xdebug
- Configuring Zend Debugger
- Validating the Configuration of a Debugging Engine

Configuring Xdebug

On this page:

- Downloading and installing the Xdebug debugging engine
- Enabling Xdebug integration with the PHP engine
- Integrating Xdebug with PhpStorm
- Configuring Xdebug for using in the On-Demand mode
- Configuring Xdebug for using in the Just-In-Time mode
  - Command-line scripts
  - Web server debugging

Downloading and installing the Xdebug debugging engine

Download the Xdebug extension compatible with your version of PHP and save it in the `php/` folder.

- The location of the `php/` folder is defined during the installation of the PHP engine .
- If you are using an AMP package , the Xdebug extension may be already installed. Follow the instructions in the `xdebug.txt` .

Enabling Xdebug integration with the PHP engine

1. Open the active `php.ini` file in the editor:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, click ... next to the Interpreter field.
   3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.
2. To disable the Zend Debugger and Zend Optimizer tools, that blocks Xdebug, remove or comment the following lines in the `php.ini` file:

```
zend_extension=<path_to_zend_debugger>
zend_extension=<path_to_zend_optimizer>
```

3. To enable Xdebug, locate the `[Xdebug]` section in the `php.ini` file and update it as follows:

```
[Xdebug]
zend_extension="<path to php_xdebug.dll>"
xdebug.remote_enable=1
xdebug.remote_port="<the port for Xdebug to listen to>" (the default port is 9000)
xdebug.profiler_enable=1
xdebug.profiler_output_dir="<AMP home\tmp>"
```

4. To enable multiuser debugging via Xdebug proxies , locate the `xdebug.idekey` setting and assign it a value of your choice. This value will be used to register your IDE on Xdebug proxy servers.
5. Save and close the `php.ini` file.

Integrating Xdebug with PhpStorm

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. Check the Xdebug installation associated with the selected PHP interpreter:
   1. On the PHP page, choose the relevant PHP installation from the Interpreter drop-down list and click the Browse button ... next to the field. The list shows all the PHP installations available in PhpStorm, see Enabling PHP Support .
   2. The Interpreters dialog box that opens shows the following:
      - The version of the selected PHP installation.
      - The name and version of the debugging engine associated with the selected PHP installation (Xdebug or Zend Debugger). If no debugger is configured, PhpStorm shows `Debugger: Not installed` .

   Alternatively, open the Xdebug checker , paste the output of the `phpinfo()` , and click Analyze my phpinfo() output.

   Learn more about checking the Xdebug installation in Validating the Configuration of a Debugging Engine .
3. Define the Xdebug behaviour. Click Debug under the PHP node. On the Debug page that opens, specify the following settings in the Xdebug area:
   - In the Debug Port text box, appoint the port through which the tool will communicate with PhpStorm. This must be exactly the same port number as specified in the `php.ini` file:

```
xdebug.remote_port = <port_number>
```

     By default, Xdebug listens on port 9000.
   - To have PhpStorm accept any incoming connections from Xdebug engines through the port specified in the Debug port text box, select the Can accept external connections check box.
   - Select the Force break at the first line when no path mapping is specified check box to have the debugger stop as soon as it reaches and opens a file that is not mapped to any file in the project on the Servers page. The debugger stops at the first line of this file and Debug Tool Window. Variables shows the following error message: Cannot find a local copy of the file on server <path to the file on the server> and a link Click to set up mappings. Click the link to open the Resolve Path

Mappings Problem dialog box and map the problem file to its local copy.
When this check box cleared, the debugger does not stop upon reaching and opening an unmapped file, the file is just processed, and no error messages are displayed.

- Select the Force break at the first line when the script is outside the project check box to have the debugger stop at the first line as soon as it reaches and opens a file outside the current project. With this check box cleared, the debugger continues upon opening a file outside the current project.

4. In the External Connections area, specify how you want PhpStorm to treat connections received from hosts and through ports that are not registered as server configurations .
   - Ignore external connections through unregistered server configurations: Select this check box to have PhpStorm ignore connections received from hosts and through ports that are not registered as server configurations . When this check box is selected, PhpStorm does not attempt to create a server configuration automatically.
   - Detect path mappings from deployment configurations:
     - When this check box is selected, PhpStorm attempts to retrieve path mappings for debugging in a remote environment from the server access configuration (deployment configuration).
     - When the check box is cleared, you have to specify the path mappings manually.

     See Validating the Configuration of a Debugging Engine , Web Server Debug Validation Dialog , and Configuring Remote PHP Interpreters for details.
   - Break at first line in PHP scripts: Select this check box to have the debugger stop as soon as connection between it and PhpStorm is established (instead of running automatically until the first breakpoint is reached). Alternatively turn on the Run | Break at first line in PHP scripts option on the main menu.
   - Max. simultaneous connections: Use this spin box to limit the number of external connections that can be processed simultaneously.

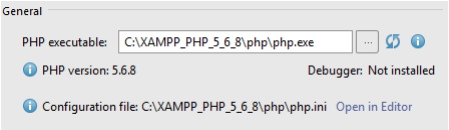
Configuring Xdebug for using in the On-Demand mode

Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable Xdebug for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster.

1. Disable Xdebug for command-line scripts:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. From the PHP executable drop-down list, choose the relevant PHP interpreter and click ... next to it. In the Interpreters dialog box that opens, click the Open in Editor link next to the Configuration file: <path to php.ini> file. Close all the dialog boxes and switch to the tab where the `php.ini` file is opened.
   3. In the `php.ini` file, find the `[XDebug]` section and comment the following line in it by adding `;` in preposition:

```
;[XDebug]

;zend_extension = "<full_path_to_xdebug>"
```

   4. Open the Interpreters dialog box and click next to the PHP executable field. PhpStorm informs you that debugger is not installed:



2. To enable PhpStorm to activate Xdebug when it is necessary, specify the path to it in the Debugger extension text box, in the Additional area. Type the path manually or click ... and select the location in the dialog box that opens.

Configuring Xdebug for using in the Just-In-Time mode

PhpStorm supports the use of Xdebug in the Just-In-Time (JIT) mode so it is not attached to your code all the time but connects to PhpStorm only when an error occurs or an exception is thrown. The Xdebug operation mode is toggled through the xdebug.remote_mode setting, which is by default set to `req` . The mode is available both for debugging command-line scripts and for web server debugging.

Depending on whether you are going to debug command-line scripts or use a Web server, use one of the scenarios below.

Command-line scripts

For debugging command-line scripts, specify the custom `-dxdebug.remote_mode=jit` directive as an additional configuration option:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. From the PHP executable drop-down list, choose the relevant PHP interpreter and click ... next to it.
3. In the Interpreters dialog box that opens, click ... next to the Configuration options text box in the Additional area.
4. In the Configuration options dialog that opens, click + to add a new entry, then type `-dxdebug.remote_mode` in the Configuration directive field and `jit` in the Value field.
   When you click OK, you return to the Interpreters dialog box where the Configuration options text box shows `-dxdebug.remote_mode=jit` .

Web server debugging

1. On the main menu, choose Run | Web Server Debug Validation.
2. In the Web Server Debug Validation Dialog that opens, choose the Web server to validate the debugger on.
   - Choose Local Web Server or Shared Folder to check a debugger associated with a local Web server.
     - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. For Web servers of the type Inplace, the folder is under the project root.
       The folder must be accessible through http.
     - URL to Validation Script: In this text box, type the URL address of the folder where the validation script will be created. If the project root is mapped to a folder

accessible through `http`, you can specify the project root or any other folder under it.

- Choose Remote Web Server to check a debugger associated with a remote server.
  - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. The folder must be accessible through http.
  - Deployment Server: In this field, specify the server access configuration of the type Local Server or Remote Server to access the target environment. For details see Configuring Synchronization with a Web Server .
    Choose a configuration from the drop-down list or click the Browse button ... in the Deployment dialog .

3. Click Validate to have PhpStorm create a validation script, deploy it to the target remote environment, and run it there.
4. Open the `php.ini` file which is reported as loaded and associated with Xdebug.
5. In the `php.ini` file, find the `[XDebug]` section and change the value of the `xdebug.remote_mode` from the default `req` to `jit`.

See also Just-In-Time debugging and PHP Exception Breakpoints with PhpStorm and Xdebug

Configuring Zend Debugger

On this page:

- Downloading and installing Zend Debugger
- Enabling Zend Debugger integration with the PHP engine
- Integrating Zend Debugger with PhpStorm
- Configuring Zend Debugger for using in the On-Demand mode

Downloading and installing Zend Debugger

1. Download the Zend Debugger package which corresponds to your operating system.
2. Locate the `ZendDebugger.so` (Unix) or `ZendDebugger.dll` (Windows) file in the directory which corresponds to your version of PHP (e.g. 4.3.x, 4.4.x, 5.0.x, 5.1.x, 5.2.x, 5.3.x, 5.4.x).
3. Copy the file to your Web server in a location that is accessible by the Web server.

Enabling Zend Debugger integration with the PHP engine

1. Open the active `php.ini` file in the editor:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, click ... next to the Interpreter field.
   3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.
2. Locate or create the `[Zend]` section.
3. To load the Zend Debugger extension, add one of the following lines inside the `[Zend]` section depending on your operating system:
   - Linux and OS X:

     ```
     zend_extension=<full_path_to_ZendDebugger.so>
     ```

   - Windows:

     ```
     zend_extension_ts=<full_path_to_ZendDebugger.dll>
     ```

   - Windows non-thread safe:

     ```
     zend_extension=<full_path_to_ZendDebugger.dll>
     ```

   Warning! The Windows non-thread safe binary file is only used with Zend Core 2.0.

4. To enable access to Zend Debugger from PhpStorm, add the following lines:

   ```
   zend_extension=<full_path_to_zend_debugger_extension>
   zend_debugger.allow_hosts=127.0.0.1
   zend_debugger.expose_remotely=allowed_host
   zend_debugger.tunnel_min_port=<any integer value above 1024>
   zend_debugger.tunnel_max_port=<any integer value below 65535>
   ```

   The value of the `zend_debugger.allow_hosts` parameter is the IPs of your machine to connect to the server debugger. It could be a comma-separated list of IPs in the format `X.X.X.X` (for example, 192.168.0.6).

   Tip For a thread-safe Windows binary use the `zend_extension_ts parameter` instead of `zend_extension`.

5. Restart your Web server.
6. To check that the Zend Debugger has been installed and configured correctly, create a file with the following contents:

   ```
   <?php
   phpinfo();
   ?>
   ```

   Open the page that corresponds to the file in the browser. The output should contain a Zend Debugger section.

Integrating Zend Debugger with PhpStorm

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. Check the Zend Debugger installation associated with the selected PHP interpreter:

1. On the PHP page, choose the relevant PHP installation from the Interpreter drop-down list and click the Browse button ... next to the field. The list shows all the PHP installations available in PhpStorm, see Enabling PHP Support .
2. The Interpreters dialog box that opens shows the following:
   - The version of the selected PHP installation.
   - The name and version of the debugging engine associated with the selected PHP installation (Xdebug or Zend Debugger). If no debugger is configured, PhpStorm shows `Debugger: Not installed` .

   Learn more about checking the Zend Debugger installation in Validating the Configuration of a Debugging Engine .
3. Define the Zend Debugger behaviour. Click Debug under the PHP node. On the Debug page that opens, specify the following settings in the Zend Debugger area:
   - In the Debug Port text box, appoint the port for PhpStorm to communicate with the tool through. Type the port number within the tunnel specified in the `php.ini` file through `zend_debugger.tunnel_min_port` and `zend_debugger.tunnel_max_port` . For details, see http://files.zend.com/help/previous-version/Zend-Server-4-Community-Edition/zenddebugger.html
   - To have PhpStorm accept any incoming connections from Zend Debugger engines through the port specified in the Debug port text box, select the Can accept external connections check box.
   - To use a debugger toolbar in the browser, specify the port through which the debugger settings are passed to the browser in the Settings broadcasting port text box.
4. In the External Connections area, specify how you want PhpStorm to treat connections received from hosts and through ports that are not registered as server configurations .
   - Ignore external connections through unregistered server configurations: Select this check box to have PhpStorm ignore connections received from hosts and through ports that are not registered as server configurations . When this check box is selected, PhpStorm does not attempt to create a server configuration automatically.
   - Detect path mappings from deployment configurations:
     - When this check box is selected, PhpStorm attempts to retrieve path mappings for debugging in a remote environment from the server access configuration (deployment configuration).
     - When the check box is cleared, you have to specify the path mappings manually.

     See Validating the Configuration of a Debugging Engine , Web Server Debug Validation Dialog , and Configuring Remote PHP Interpreters for details.
   - Break at first line in PHP scripts: Select this check box to have the debugger stop as soon as connection between it and PhpStorm is established (instead of running automatically until the first breakpoint is reached). Alternatively turn on the Run | Break at first line in PHP scripts option on the main menu.
   - Max. simultaneous connections: Use this spin box to limit the number of external connections that can be processed simultaneously.
5. To block requests from the Z-Ray system if they annoy you by invoking the PhpStorm debugger too often, select the Ignore Z-Ray system requests check box.

Configuring Zend Debugger for using in the On-Demand mode

Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable Zend Debugger for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster.

1. Disable Zend Debugger for command-line scripts:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. From the PHP executable drop-down list, choose the relevant PHP interpreter and click ... next to it. In the Interpreters dialog box that opens, click the Open in Editor link next to the Configuration file: <path to php.ini> file. Close all the dialog boxes and switch to the tab where the `php.ini` file is opened.
   3. In the `php.ini` file, find the `[Zend]` section and comment the following lines in it by adding `;` in preposition: Linux and OS X:

      ```
      zend_extension=<full_path_to_ZendDebugger.so>
      ```

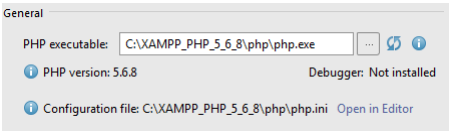
      Windows:

      ```
      zend_extension_ts=<full_path_to_ZendDebugger.dll>
      ```

      Windows non-thread safe:

      ```
      zend_extension=<full_path_to_ZendDebugger.dll>
      ```

      Warning! The Windows non-thread safe binary file is only used with Zend Core 2.0.
   4. Open the Interpreters dialog box and click next to the PHP executable field. PhpStorm informs you that debugger is not installed:

2. To enable PhpStorm to activate Zend Debugger when it is necessary, specify the path to it in the Debugger extension text box, in the Additional area. Type the path manually or click ... and select the location in the dialog box that opens.

Validating the Configuration of a Debugging Engine

PhpStorm can validate your configuration of Xdebug or Zend Debugger and tell you if some setting is missing (for example, `xdebug.remote_enable`) or inconsistent with other settings. When configuring the PHP interpreter for a project, PhpStorm informs you whether a debugger is installed in your local PHP development environment and reports on the Xdebug or Zend Debugger version used. For details, see Configuring a Debugging Engine and Enabling PHP Support .

You can also get more detailed information about the debugging engine on a local or remote Web server.

1. On the main menu, choose Run | Web Server Debug Validation.
2. In the Web Server Debug Validation Dialog that opens, choose the Web server to validate the debugger on.
   - Choose Local Web Server or Shared Folder to check a debugger associated with a local Web server.
     - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. For Web servers of the type Inplace, the folder is under the project root.
       The folder must be accessible through http.
     - URL to Validation Script: In this text box, type the URL address of the folder where the validation script will be created. If the project root is mapped to a folder accessible through `http`, you can specify the project root or any other folder under it.
   - Choose Remote Web Server to check a debugger associated with a remote server.
     - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. The folder must be accessible through http.
     - Deployment Server: In this field, specify the server access configuration of the type Local Server or Remote Server to access the target environment. For details see Configuring Synchronization with a Web Server .
       Choose a configuration from the drop-down list or click the Browse button ... in the Deployment dialog .
3. Click Validate to have PhpStorm create a validation script, deploy it to the target remote environment, and run it there.

PHP Debugging Session

Before you start debugging, make sure that you have a debugging engine installed and configured properly. PhpStorm supports debugging with two most popular tools: Xdebug and Zend Debugger . These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file as described in Configuring Xdebug and Configuring Zend Debugger .

To open the active `php.ini` file in the editor:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click ... next to the Interpreter field.
3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.

For more information on configuring debugging engines, see Xdebug Installation Guide and Zend Debugger Installation Guide .

With PhpStorm, a PHP debugging session can be initiated either through a debug configuration or without it. The latter approach is also called Zero-configuration debugging. PhpStorm supports three main ways to initiate a PHP debugging session:

- You create a PHP Web Application debug configuration, and then PhpStorm uses its settings to launch the application, open the browser, and activate the debugging engine.
- You create a PHP HTTP Request debug configuration, PhpStorm generates a request on its base, and then accesses a specific page through this request.
- Zero-configuration debugging, when no debug configuration is created at all. Instead, you open the starting page of your PHP application in the browser manually and then activate the debugging engine from the browser, while PhpStorm listens to incoming debugger connections.

No matter which method you choose, you can specify the scripts requests to which you want PhpStorm to ignore during debugging. This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

When using Xdebug, you can also debug PHP applications in the multiuser mode via Xdebug proxy servers.

In this part:

- Creating a PHP Debug Server Configuration
- Debugging with a PHP Web Application Debug Configuration
- Zero-Configuration Debugging
- Debugging a PHP HTTP Request
- Debugging in the Just-in-Time Mode
- Debugging with PHP Exception Breakpoints

Creating a PHP Debug Server Configuration

The term PHP debug server configuration denotes the settings that ensure HTTP access for debugging engines to interact with local and remote Web servers and set correspondence between files on the server and their local copies in the PhpStorm project. The settings from debug server configurations are used when debugging with run/debug configurations of the type PHP Web Application or PHP Remote Debug and during Zero-Configuration Debugging sessions.

You can create a debug server configuration manually from scratch or import some settings from a server access (deployment) configuration (see Configuring Synchronization with a Web Server for details).

On this page:

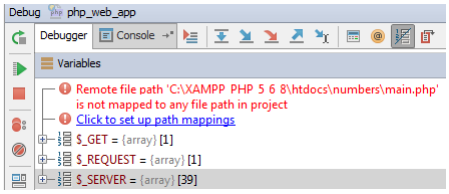
- Defining a debug server configuration manually
- Importing settings from a server access (deployment) configuration

Defining a debug server configuration manually

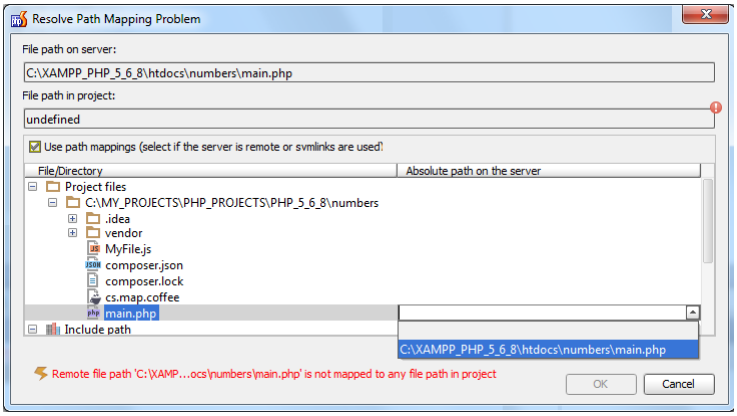
1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Servers under PHP.
   Alternatively, click ... next to the Server drop-down list in the Run/Debug Configuration: PHP Web Application dialog box.

   In either case, the Servers page opens with the left-hand pane showing all the available debug server configurations. Click + on the toolbar and specify the debug server settings in the right-hand pane.
2. Specify the debug server configuration name.
3. Specify the host where the application is run and the port to access it.
4. From the Debugger drop-down list, choose the debugging engine to use.
5. Specify how PhpStorm will set up correspondence between files on the server and their local copies. Based on these mappings, PhpStorm will open local copies of currently processed files.
   Path mappings in PHP Debug Server configurations look very similar to the path mappings in server access (deployment configurations. Unfortunately, they cannot be reused, as deployment configurations uses relative paths while PHP Debug Servers configurations rely on absolute paths.
   - Select the Use path mappings check box if you are working on a remote Web server, that is, when the Web server is on a physically remote host, or the Web server is installed on your machine but your project is outside the Web server document root. Also select the check box if you are using symlinks.
     Map the absolute paths to the files and folders on the server with absolute paths to your project files in the local file system using the Path on server and File/Directory fields respectively.
     - File / Directory: This read-only field displays the files and folders of the current project. Select a file or a folder to be used as the local copy.
     - Path on server: In this filed, specify the absolute path to the file or folder on the target server to which the selected local file or folder corresponds. Type the path manually or select it from the drop-down list.
   - Clear the Use path mappings check box if you are working right on your Web server so your project root is under the server document root. In this case the absolute paths to the files on the Web server and the absolute paths to the corresponding files in your project are the same.

   If you do not specify any path mappings and start debugging an application that is not under the server document root, PhpStorm displays an error message:



   The Click to set up path mappings link brings up the Resolve Path Mappings Problem dialog box, where you can define the path mappings:



   When you click OK and leave the dialog box, PhpStorm selects the Use path mappings check box on the Servers page automatically.
6. Select the Shared check box to share the debug server configuration across a team. The host/port settings and the path mappings are stored in the `.idea/php.xml` file is available to all team members through a version control system. Note that mappings are shared only for directories inside the project.

Importing settings from a server access (deployment) configuration

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Servers under PHP.
   Alternatively, click ... next to the Server drop-down list in the Run/Debug Configuration: PHP Web Application dialog box.

   In either case, the Servers page opens with the left-hand pane showing all the available debug server configurations. Click on the toolbar. The Import from Deployment Configuration Dialog dialog box opens.

2. From the Deployment drop-down list, choose the server access configuration (deployment configuration) to copy the server access settings from. The list contains all the available deployment configurations. To create a new configuration, click and specify new settings in the Deployment: Connection Tab dialog box that opens.
3. Depending on the type of the server access configuration chosen from the Deployment drop-down list, specify one of the following:
   - For an FTP, SFTP, or FTPS server access configuration, specify the absolute path to the server deployment root. This path will be added as a prefix to the path from the Root Path text box on the Deployment: Connection Tab .

     If you are not sure about this absolute path, you can open the Remote Host tool window, choose the required deployment configuration, position the cursor at the root folder, and choose Copy Path on the context menu, see Accessing Files on Web Servers for details. Alternatively, contact your hosting provider.
   - For Local or mounted folder, type the absolute path to the server root as specified in the Folder field of the Creating a PHP Debug Server Configuration dialog box.
   - For Inplace Server configurations no mappings are required because the local and remote paths are the same in this case.
4. The Preview area shows the host/port and the path mappings retrieved from the chosen server access configuration (deployment configuration).

   When you choose the deployment configuration to use, the Absolute path on the server field shows relative paths mapped to the project files and folders in the chosen configuration, that is, paths to files and folders relative to the deployment root. As you specify the absolute path to the deployment root (the server root for FTP/SFTP/FTPS or the mounted folder), the contents of the field are updated automatically and finally the field shows absolute paths on the server.

Debugging with a PHP Web Application Debug Configuration

On this page:

- Introduction
- Preparing the debugging engine
- Setting breakpoints
- Creating a debug configuration of the type PHP Web Application
- Initiating a debugging session and examining the suspended program
- Specifying scripts to skip requests to

Introduction

In this debugging mode, PhpStorm fully controls the debugging process: it launches the application, opens the browser, and activates the debugging engine according to a PHP Web Application debug configuration.

A PHP Web Application debug configuration tells PhpStorm the URL address to access the starting page of the application, the browser to open the starting page in, and the debug server configuration to use.

You can also specify the scripts requests to which you want PhpStorm to ignore during debugging. This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

Preparing the debugging engine

Before you start debugging, make sure that you have a debugging engine installed and configured properly. PhpStorm supports debugging with two most popular tools: Xdebug and Zend Debugger . These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file as described in Configuring Xdebug and Configuring Zend Debugger .

To open the active `php.ini` file in the editor:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click ... next to the Interpreter field.
3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.

For more information on configuring debugging engines, see Xdebug Installation Guide and Zend Debugger Installation Guide .

Setting breakpoints

Breakpoints are source code markers used to trigger actions during a debugging session. Typically, the purpose behind setting a breakpoint is to suspend program execution to allow you to examine program data. However, PhpStorm can use breakpoints as triggers for a variety of different actions. Breakpoints can be set at any time during the debugging process. Your breakpoints don't affect your source files directly, but the breakpoints and their settings are saved with your PhpStorm project so you can reuse them across debugging sessions.

1. Place the caret on the desired line of the source code.
   Breakpoints can be set in the PHP context inside `*.php` , `*.html` , and files of other types. Only executable lines are valid locations for line breakpoints. Comments, declarations, and empty lines are not valid locations for the
2. Do one of the following:
   - Click the left gutter area at a line where you want to toggle a breakpoint.
   - On the main menu, choose Run | Toggle Line Breakpoint.
   - Press `Ctrl+F8` .

Creating a debug configuration of the type PHP Web Application

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Press `Shift+Alt+F10` , then press `0` to display the Edit Configuration dialog box or select the configuration from the pop-up window and press `F4` .
2. Click + on the toolbar or press `Insert` . From the drop-down list, select the PHP Web Application configuration type. The PHP Web Application dialog box opens.
3. Specify the configuration name.
4. Choose the applicable debug server configuration from the Server drop-down list or click the Browse button ... and define a debug server configuration in the Servers dialog box that opens as described in Creating a PHP Debug Server Configuration
5. In the Start URL text box, type the server path to the file that implements the application starting page. Specify the path relative to the server configuration root (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.). The read-only field below shows the URL address of the application starting page. The URL address is composed dynamically as you type.
6. Specify the browser to open the application in. Choose a configured browser from the Browser drop-down list or click the Browse button ... and specify another browser in the Web Browsers dialog box that opens.

Initiating a debugging session and examining the suspended program

1. To start debugging, click the Debug button on the toolbar.
2. As soon as the debugger suspends on reaching the first breakpoint, examine the application by analyzing frames. A frame corresponds to an active method or function call and stores the local variables of the called method or function, the arguments to it, and the code context that enables expression evaluation. All currently active frames are displayed on the Frames pane of the Debug tool window . where you can switch between them and analyze the information stored therein in the Variables and Watches panes. For more details, see the section Examining Suspended Program .
3. Continue running the program and examine its frames as soon as it is suspended again.
   - To control the program execution manually, step through the code using the commands under the Run menu or toolbar buttons: Step Into (`F7`), Step Out (`Shift+F8`), Step Over (`F8`), and others. For more details, see Stepping Through the Program .

- To have the program run automatically up to the next breakpoint, resume the session by choosing Run | Resume Program or pressing `F9`

Specifying scripts to skip requests to

This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the PHP node under Languages & Frameworks, and then click Skipped Paths under Debug.
2. On the Skipped Paths page that opens, configure an "ignore list" of scripts and folders with scripts not to be invoked if PhpStorm receives incoming connections to them.
   - To add a new entry to the list, click the Add button + or press `Alt+Insert`. Then click the Browse button and in the dialog box that opens choose the file or folder to skip connections to.
   - To remove an entry from the list, select it and click the Remove button — or press `Alt+Delete`. The script will be now executed upon receiving requests to it.
3. To have PhpStorm inform you every time it receives a request to a script to be skipped, select the Notify about skipped paths check box.

Zero-Configuration Debugging

On this page:

- Overview
- Preparing the debugging engine
- Setting the breakpoints
- Generating the Start Debugger/Stop Debugger bookmarklets
- Initiating a debugging eession and examining the suspended program
- Troubleshooting
  - No Path Mappings are Configured
  - No Debug Server is Configured
  - A Script Is not Suspended
- Specifying the scripts to skip requests to
- Starting a debugging session from the command line
  - Starting a Script with Xdebug
  - Starting a script with Zend Debugger
  - Configuring path mappings
  - Troubleshooting

Overview

In case of zero-configuration debugging, you do not need to create any debug configuration. Instead, you open the starting page of your PHP application in the browser manually, and then activate the debugging engine from the browser, while PhpStorm listens to incoming debugger connections.

To enable starting and stopping the debugging engine from the browser manually, you need to set a special `GET` / `POST` or `COOKIE` parameter. You can do it manually in the `php.ini` configuration file or generate the Start Debugger/ Stop Debugger bookmarklets on the toolbar of your browser. These bookmarklets provide control over the debugger cookie, through them you will activate and deactivate the debugger.

For more details about setting the parameters manually, see Starting the Debugger for Xdebug and Zend Debugger GET Request Parameters for Zend Debugger.

You can also specify the scripts requests to which you want PhpStorm to ignore during debugging. This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

Preparing the debugging engine

Before you start debugging, make sure that you have a debugging engine installed and configured properly. PhpStorm supports debugging with two most popular tools: Xdebug and Zend Debugger . These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file as described in Configuring Xdebug and Configuring Zend Debugger .

To open the active `php.ini` file in the editor:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click ... next to the Interpreter field.
3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.

For more information on configuring debugging engines, see Xdebug Installation Guide and Zend Debugger Installation Guide .

Setting the breakpoints

Breakpoints are source code markers used to trigger actions during a debugging session. Typically, the purpose behind setting a breakpoint is to suspend program execution to allow you to examine program data. However, PhpStorm can use breakpoints as triggers for a variety of different actions. Breakpoints can be set at any time during the debugging process. Your breakpoints don't affect your source files directly, but the breakpoints and their settings are saved with your PhpStorm project so you can reuse them across debugging sessions.

1. Place the caret on the desired line of the source code.
   Breakpoints can be set in the PHP context inside `*.php` , `*.html` , and files of other types. Only executable lines are valid locations for line breakpoints. Comments, declarations, and empty lines are not valid locations for the
2. Do one of the following:
   - Click the left gutter area at a line where you want to toggle a breakpoint.
   - On the main menu, choose Run | Toggle Line Breakpoint.
   - Press `Ctrl+F8` .

Generating the Start Debugger/Stop Debugger bookmarklets

These bookmarklets will appear on the toolbar of your browser. They provide control over the debugger cookie, through them you will activate and deactivate the debugger.

1. Enable the Bookmarks toolbar in your browser by doing one of the following depending on the browser type:
   - In Firefox, choose View | Toolbar | Bookmarks Toolbar.
   - In Chrome, choose Bookmarks | Show bookmarks bar.
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Debug under PHP.
3. On the Debug page, that opens, click the Use debugger bookmarklets to initiate debugger from your favorite browser link.
4. On the Zend Debugger & Xdebug bookmarklets page that opens, check the debugging engine settings and click Generate. The bookmarks for listed debugging-related actions are generated.
5. Drag the generated links to the bookmark toolbar in your browser.

Initiating a debugging eession and examining the suspended program

1. Generate the bookmarklets to toggle the debugger through. These bookmarklets will appear on the toolbar of your browser. They provide control over the debugger cookie, through them you will activate and deactivate the debugger.
   1. Enable the Bookmarks toolbar in your browser by doing one of the following depending on the browser type:
      - In Firefox, choose View | Toolbar | Bookmarks Toolbar.
      - In Chrome, choose Bookmarks | Show bookmarks bar.
   2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Debug under PHP.
   3. On the Debug page, that opens, click the Use debugger bookmarklets to initiate debugger from your favorite browser link.
   4. On the Zend Debugger & Xdebug bookmarklets page that opens, check the debugging engine settings and click Generate. The bookmarks for listed debugging-related actions are generated.
   5. Drag the generated links to the bookmark toolbar in your browser.
2. Set the breakpoints , where necessary.
3. Toggle the Start Listen PHP Debug Connections button so it changes to . After that PhpStorm starts listening to the port of the debugging engine used in the current project . Ports for debuggers are set at the PhpStorm level in the Debug dialog box (File | Settings | PHP | Debug for Windows and Linux or PhpStorm | Preferences | PHP | Debug for OS X).
4. Open the starting page of your application in the browser, choose the Start debugger bookmark to activate the debugging engine from the browser, re-load the current page (the starting page of the application), and then return to PhpStorm.
5. As soon as the debugger suspends on reaching the first breakpoint, examine the application by analyzing frames. A frame corresponds to an active method or function call and stores the local variables of the called method or function, the arguments to it, and the code context that enables expression evaluation. All currently active frames are displayed on the Frames pane of the Debug tool window . where you can switch between them and analyze the information stored therein in the Variables and Watches panes. For more details, see the section Examining Suspended Program .
6. Continue running the program and examine its frames as soon as it is suspended again.
   - To control the program execution manually, step through the code using the commands under the Run menu or toolbar buttons: Step Into (`F7`), Step Out (`Shift+F8`), Step Over (`F8`), and others. For more details, see Stepping Through the Program .
   - To have the program run automatically up to the next breakpoint, resume the session by choosing Run | Resume Program or pressing `F9`

Troubleshooting

During a debugging session you may encounter one of the following problems:

No Path Mappings are Configured

In some cases you may get one of the following error messages: Remote file path path/to/script/on/the/server.php is not mapped to any file path in project or The script path/to/script/on/the/server.php is outside the project. This means that PhpStorm is not sure which local file corresponds to the specified remote file path because you have not specified the path mappings in the Debug server configuration.

Even if path mappings in a Debug Server configuration look very similar to path mappings in a Deployment configuration, they cannot be reused, because Deployment configurations require paths to be relative, and Debug Server configurations rely on absolute paths.

Click the link and configure path mappings in the dialog box that opens, as described in Creating a PHP Debug Server Configuration .

No Debug Server is Configured

If you initiate a debugging session without having configured a Debug server, upon establishing connection PhpStorm displays the Incoming Connection Dialog where suggests importing mappings from a server access configuration (deployment configuration). If you choose Import mappings from deployment, PhpStorm tries to detect the most suitable deployment configuration, preselects it in the Deployment drop-down list, and the Preview area shows the absolute path to the project file which corresponds to the currently executed script according to the mappings from the selected configuration.

If PhpStorm does not detect a relevant configuration:

1. Choose the most suitable configuration from the drop-down list or click and create a new configuration in the Deployment dialog box that opens, whereupon the new configuration is added to the list.
2. In the Deployment root text box, type the absolute path to the server root folder

You can also select the Manually choose local file or project option, in this case PhpStorm displays the project tree view where you can select a project file and map the currently executed script to it. You can also select and map the entire project.

To continue debugging with the imported or manually specified configuration settings, click Accept.

A Script Is not Suspended

Establishing a Zero-Configuration debugging session may fail, with no breakpoints hit and therefore the script not suspended. This may happen if if the path mappings are not configured or configured erroneously, or if you have not set any breakpoints. In the latter case, enabling selecting the Break at First Line in PHP Scripts check box in the External Connections area or turning the Run | Break at First Line in PHP Scripts option on the main menu may also help.

To have PhpStorm display a notification if the script is not suspended, select the Notify if debug session was finished without being stopped check box in the Advanced Settings area on the Debug page of the Settings dialog box.

Specifying the scripts to skip requests to

This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the PHP node under Languages & Frameworks, and then click Skipped Paths under Debug.
2. On the Skipped Paths page that opens, configure an "ignore list" of scripts and folders with scripts not to be invoked if PhpStorm receives incoming connections to them.
   - To add a new entry to the list, click the Add button + or press `Alt+Insert`. Then click the Browse button and in the dialog box that opens choose the file or

folder to skip connections to.

- To remove an entry from the list, select it and click the Remove button — or press `Alt+Delete`. The script will be now executed upon receiving requests to it.

3. To have PhpStorm inform you every time it receives a request to a script to be skipped, select the Notify about skipped paths check box.

Starting a debugging session from the command line

You can start debugging a PHP CLI script from the command line, having PhpStorm listen for incoming debugger connections.

1. Set the breakpoints where necessary.
2. On the toolbar, toggle the Start Listening for PHP Debug Connections button or choose Run | Start Listening for PHP Debug Connections on the main menu.
3. Start the script with debugger options depending on the debugging engine you are using - Xdebug or Zend Debugger.

Starting a Script with Xdebug

Xdebug has various configuration options which can be used to let the PHP interpreter reach out to PhpStorm. These parameters have to be passed to the PHP interpreter using the - `d` command line switch. A more convenient is to set an environment variable you do not need to provide the `-d` switches all the time.

Do one of the following:

- Launch PHP with several switches:

```
php -dxdebug.remote_enable=1 -dxdebug.remote_mode=req -dxdebug.remote_port=9000 -dxdebug.remote_host=127.0.0.1 -dxdebug.remote_connect_back=0 p
```

- Set an environment variable that configures Xdebug:
  - `set XDEBUG_CONFIG="remote_enable=1 remote_mode=req remote_port=9000 remote_host=127.0.0.1 remote_connect_back=0"` for Windows
  - `export XDEBUG_CONFIG="remote_enable=1 remote_mode=req remote_port=9000 remote_host=127.0.0.1 remote_connect_back=0"` for Linux and OS X

Starting a script with Zend Debugger

Zend Debugger has various configuration options which can be used to let the PHP interpreter reach out to PhpStorm. These parameters have to be passed to the PHP interpreter using an environment variable:

- `set QUERY_STRING="start_debug=1&debug_host=127.0.0.1&no_remote=1&debug_port=10137&debug_stop=1"` for Windows
- `export QUERY_STRING="start_debug=1&debug_host=127.0.0.1&no_remote=1&debug_port=10137&debug_stop=1"` for Linux and OS X

Configuring path mappings

To tell PhpStorm which path mapping configuration should be used for a connection from a certain machine, set the value of the `PHP_IDE_CONFIG` environment variable to `serverName=SomeName`, where `SomeName` is the name of the debug server configuration defined on the Servers page, see Creating a PHP Debug Server Configuration. Depending on the operating system you are using, set the value in one of the following formats:

- `set PHP_IDE_CONFIG="serverName=SomeName"` for Windows
- `export PHP_IDE_CONFIG="serverName=SomeName"` for Linux and OS X

Troubleshooting

- The debug server configuration is not specified through the `PHP_IDE_CONFIG` environmanet variable. In this case, PhpStorm detects the host and port from `$_SERVER['SSH_CONNECTION']` and suggests to create a new debug server configuration if it doesn't exists. This works for all connections through ssh even without a tunnel.
- The debug server configuration is not specified through the `PHP_IDE_CONFIG` environment variable and `$_SERVER['SSH_CONNECTION']` is not defined. In this case, a warning is displayed with a link to the instruction in specifying the debug server configuration through the `PHP_IDE_CONFIG` environment variable.
- The debug server configuration is specified through the `PHP_IDE_CONFIG` environment variable but a wrong format is used, PhpStorm displays an error message with the instructions.
- The `PHP_IDE_CONFIG` environment variable is configured properly, but the specified debug server configuration does not exist, PhpStorm displays a warning with a link to the Servers page.

Debugging a PHP HTTP Request

Besides debugging the entire application, you can debug separate HTTP Requests . This is helpful when you are actually interested in a specific page that is accessed in a number of steps, but for this or that reason you cannot specify this page as the start page for debugging, for example, because you need to "come" to this page with certain data.

Debugging PHP HTTP requests in PhpStorm is supported through the PHP HTTP Request run configuration. Based on the configuration settings, PhpStorm composes the request to run.

On this page:

- Preparing the debugging engine
- Setting the breakpoints
- Creating a debug configuration of the type PHP HTTP Request
- Initiating a debugging session and examining the suspended program

Preparing the debugging engine

Before you start debugging, make sure that you have a debugging engine installed and configured properly. PhpStorm supports debugging with two most popular tools: Xdebug and Zend Debugger . These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file as described in Configuring Xdebug and Configuring Zend Debugger .

To open the active `php.ini` file in the editor:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. On the PHP page that opens, click ... next to the Interpreter field.
3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.

For more information on configuring debugging engines, see Xdebug Installation Guide and Zend Debugger Installation Guide .

Setting the breakpoints

Breakpoints are source code markers used to trigger actions during a debugging session. Typically, the purpose behind setting a breakpoint is to suspend program execution to allow you to examine program data. However, PhpStorm can use breakpoints as triggers for a variety of different actions. Breakpoints can be set at any time during the debugging process. Your breakpoints don't affect your source files directly, but the breakpoints and their settings are saved with your PhpStorm project so you can reuse them across debugging sessions.

1. Place the caret on the desired line of the source code.
   Breakpoints can be set in the PHP context inside `*.php` , `*.html` , and files of other types. Only executable lines are valid locations for line breakpoints. Comments, declarations, and empty lines are not valid locations for the
2. Do one of the following:
   - Click the left gutter area at a line where you want to toggle a breakpoint.
   - On the main menu, choose Run | Toggle Line Breakpoint.
   - Press `Ctrl+F8`.

Creating a debug configuration of the type PHP HTTP Request

PhpStorm agglutinates the settings specified in this configuration into a PHP HTTP request.

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Press `Shift+Alt+F10`, then press `0` to display the Edit Configuration dialog box or select the configuration from the pop-up window and press `F4`.
2. Click + on the toolbar or press `Insert`. From the drop-down list, select the PHP HTTP Request configuration type. The PHP HTTP Request dialog box opens.
3. Specify the configuration name.
4. In the Server drop-down list, specify the debug server configuration to interact with the Web server where the application is executed. Select one of the existing configurations or click the Browse button ... and define a debug server configuration in the Servers dialog box that opens as described in Creating a PHP Debug Server Configuration .
5. In the URL text box, complete the `host` element of the request to debug. Type the path relative to the host specified in the debug server configuration. As you type, PhpStorm composes the URL address on-the-fly and displays it below the text box.
6. Specify whether you want to bring any data to the target page. From the Request method drop-down list, choose the relevant request type:
   - To access the page without bringing any data, choose GET.
   - To access the page with some data saved in variables, choose POST and type the relevant variables in the Request body text box.
     By default, the Project Encoding is used in requests encoding if it is not specified explicitly, for example:

     ```
     header('Content-type: text/html;charset=utf-8');
     ```

     The Project Encoding is specified on the File Encodings page, under the Editor node of the Settings / Preferences Dialog .
7. In the Query text box, type the query string of the request. This string will be appended to the request after the `?` symbol.
8. Click OK, when ready.

Initiating a debugging session and examining the suspended program

1. To start debugging, click the Debug button on the toolbar.
2. As soon as the debugger suspends on reaching the first breakpoint, examine the application by analyzing frames. A frame corresponds to an active method or function call and stores the local variables of the called method or function, the arguments to it, and the code context that enables expression evaluation. All currently active frames are displayed on the Frames pane of the Debug tool window . where you can switch between them and analyze the information stored therein in the Variables and Watches panes. For more details, see the section Examining Suspended Program .

3. Continue running the program and examine its frames as soon as it is suspended again.
   - To control the program execution manually, step through the code using the commands under the Run menu or toolbar buttons: Step Into (`F7`), Step Out (`Shift+F8`), Step Over (`F8`), and others. For more details, see Stepping Through the Program .
   - To have the program run automatically up to the next breakpoint, resume the session by choosing Run | Resume Program or pressing `F9`

Debugging in the Just-In-Time Mode

PhpStorm supports the use of Xdebug in the Just-In-Time (JIT) mode so it is not attached to your code all the time but connects to PhpStorm only when an error occurs or an exception is thrown. The Xdebug operation mode is toggled through the xdebug.remote_mode setting, which is by default set to `req`. The mode is available both for debugging command-line scripts and for web server debugging.

On this page:

- Configuring Xdebug for using in the Just-In-Time mode
  - Command-line scripts
  - Web server debugging
- Debugging session

Configuring Xdebug for using in the Just-In-Time mode

Depending on whether you are going to debug command-line scripts or use a Web server, use one of the scenarios below.

Command-line scripts

For debugging command-line scripts, specify the custom `-dxdebug.remote_mode=jit` directive as an additional configuration option:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
2. From the PHP executable drop-down list, choose the relevant PHP interpreter and click ... next to it.
3. In the Interpreters dialog box that opens, click ... next to the Configuration options text box in the Additional area.
4. In the Configuration options dialog that opens, click + to add a new entry, then type `-dxdebug.remote_mode` in the Configuration directive field and `jit` in the Value field.
   When you click OK, you return to the Interpreters dialog box where the Configuration options text box shows `-dxdebug.remote_mode=jit`.

Web server debugging

1. On the main menu, choose Run | Web Server Debug Validation.
2. In the Web Server Debug Validation Dialog that opens, choose the Web server to validate the debugger on.
   - Choose Local Web Server or Shared Folder to check a debugger associated with a local Web server.
     - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. For Web servers of the type Inplace, the folder is under the project root.
       The folder must be accessible through http.
     - URL to Validation Script: In this text box, type the URL address of the folder where the validation script will be created. If the project root is mapped to a folder accessible through `http`, you can specify the project root or any other folder under it.
   - Choose Remote Web Server to check a debugger associated with a remote server.
     - Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. The folder must be accessible through http.
     - Deployment Server: In this field, specify the server access configuration of the type Local Server or Remote Server to access the target environment. For details see Configuring Synchronization with a Web Server.
       Choose a configuration from the drop-down list or click the Browse button ... in the Deployment dialog.
3. Click Validate to have PhpStorm create a validation script, deploy it to the target remote environment, and run it there.
4. Open the `php.ini` file which is reported as loaded and associated with Xdebug.
5. In the `php.ini` file, find the `[XDebug]` section and change the value of the `xdebug.remote_mode` from the default `req` to `jit`.

Debugging session

Set the breakpoints and launch a debugging session, as described in Initiating a Debugging Session or Debugging with a PHP Web Application Debug Configuration.

Xdebug connects to PhpStorm in the following two cases:

- When an error occurs. In this case, Xdebug stops on the line right after the error condition. The reason for that is that PhpStorm first has to run the erroneous code before it knows something is wrong.
- When an exception is thrown. If the exception is handled, Xdebug breaks at the first line of the `catch` block if there is one, or at the `finally` block.

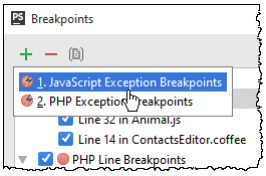
Debugging with PHP Exception Breakpoints

On this page:

- Configuring PHP Exception breakpoints
- Examining the suspended program

Configuring PHP Exception breakpoints

1. On the main menu, choose Run | View Breakpoints, or press `Ctrl+Shift+F8`.
2. In the Breakpoints dialog box that opens, click +.
3. From the drop-down list, choose PHP Exception Breakpoint.



4. In the Add Exception Breakpoint dialog box that opens, specify the errors or exceptions on which you want the debugger to suspend.
   - To break on PHP error conditions, choose one of the standard types from the drop-down list, the available options are Warning, Notice, or Deprecated.
   - Alternatively, specify a custom Exception type. Note that E_ERROR, E_PARSE, and E_COMPILE_ERROR are not handled as they halt execution of the PHP engine.

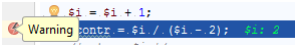
   Click OK when ready.

   PhpStorm returns you to the Breakpoints dialog box.
5. Configure the new exception breakpoint as described in Configuring Breakpoints .

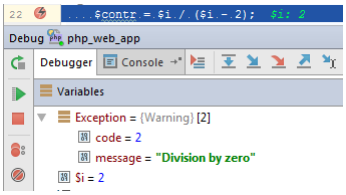
Examining the suspended program

When the debugger breaks on an error or an exception, PhpStorm sets a PHP Exception Breakpoint.

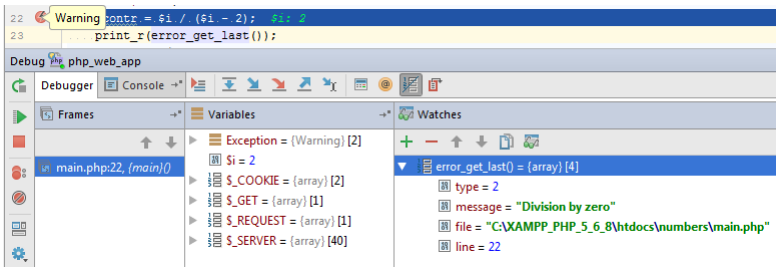
- To see the breakpoint type, hover the mouse pointer over the breakpoint. The type is displayed in a pop-up window:



- The Variables pane displays a fake `Exception` variable which shows the exception message and the exception code:



- To get more information on PHP errors, add a watch for the `error_get_last()` function, see Adding, Editing and Removing Watches . Then the details of errors will be displayed in the Watches pane:



Multiuser Debugging via Xdebug Proxies

When using Xdebug, you can debug PHP applications in the multiuser mode via Xdebug proxy servers.

To enable multiuser debugging via an Xdebug proxy server

1. Open the active `php.ini` file in the editor:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, click ... next to the Interpreter field.
   3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.
2. Locate the `xdebug.idekey` setting and assign it a value of your choice.
3. Configure access to the required Xdebug proxy server from your IDE:
   - To register your IDE to the server choose Tools | Xdebug Proxy | Register IDE on the main menu. In the Xdebug Proxy dialog box, that opens, specify the `xdebug.idekey`, the host on which the Xdebug proxy server resides, and the port which PhpStorm will listen to during a proxy debugging session. When you click OK, PhpStorm connects to the specified proxy server and sends the specified credentials. The server registers the credentials and sends a confirmation.
   - To update the credentials, choose Tools | Xdebug Proxy | Configuration on the main menu. In the Xdebug Proxy dialog box, that opens, edit the IDE key, the host, and the port settings.
   - To discard the current credentials, choose Tools | Xdebug Proxy | Cancel IDE Registration on the main menu.

Profiling the Performance of a PHP Application

Besides pure debugging, you can also profile the performance of your applications. PhpStorm provides visual representation of the profiling data collected by the debugging engine you are currently using.

In this section:

- Profiling with Xdebug
  - Enabling Profiling with Xdebug
  - Analyzing Xdebug Profiling Data
- Profiling with Zend Debugger
  - Enabling Profiling with Zend Debugger
  - Analyzing Zend Debugger Profiling Data

Profiling with Xdebug

Besides interactive debugging , PhpStorm integration with Xdebug also supports profiling. PhpStorm provides visual representation of profiling data generated by Xdebug. You can select several snapshots at a time and collect the aggregated profiling information.

Before profiling with Xdebug, download, install and configure the components of the PHP development environment . Normally, these are a PHP engine, a web server, and the Xdebug tool.

In this part:

- Enabling Profiling with Xdebug
- Analyzing Xdebug Profiling Data

Enabling Profiling with Xdebug

Xdebug profiler is incorporated in the Xdebug tool. Therefore you first need to download, install, and enable Xdebug itself and after that enable the profiling functionality within it.

To enable profiling with Xdebug, perform these general steps:

- Configuring Xdebug
- Enabling the Xdebug profiler
- Configuring the way to toggle the profiler from the browser
- Specifying the location for storing accumulated profiling data

Configuring Xdebug

1. Download and install the Xdebug tool.
2. Integrate Xdebug with the PHP engine .
3. Integrate Xdebug with PhpStorm .

Enabling the Xdebug profiler

1. Open the active `php.ini` file in the editor:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, click ... next to the Interpreter field.
   3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.
2. Set the xdebug.profiler_enable directive to 1:

```
xdebug.profiler_enable = 1;
```

3. To enable toggling the profiler from the browser through control over debugger cookies, set the xdebug.profiler_enable_trigger directive to 1:

```
xdebug.profiler_enable_trigger = 1;
```

Configuring the way to toggle the profiler from the browser

To specify the GET/POST or COOKIE parameters, do one of the following:

- Specify the values manually .
- Generate the bookmarklets to toggle the debugger through. These bookmarklets will appear on the toolbar of your browser. They provide control over the debugger cookie, through them you will activate and deactivate the debugger.
  1. Enable the Bookmarks toolbar in your browser by doing one of the following depending on the browser type:
     - In Firefox, choose View | Toolbar | Bookmarks Toolbar.
     - In Chrome, choose Bookmarks | Show bookmarks bar.
  2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Debug under PHP.
  3. On the Debug page, that opens, click the Use debugger bookmarklets to initiate debugger from your favorite browser link.
  4. On the Zend Debugger & Xdebug bookmarklets page that opens, check the debugging engine settings and click Generate. The bookmarks for listed debugging-related actions are generated.
  5. Drag the generated links to the bookmark toolbar in your browser.

Specifying the location for storing accumulated profiling data

1. Open the active `php.ini` file in the editor:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, click ... next to the Interpreter field.
   3. In the Interpreters dialog box that opens, the Configuration File read-only field shows the path to the active `php.ini` file. Click Open in Editor.
2. Define location for accumulating profiling snapshots by specifying the xdebug.profiler_output_dir directive.

```
xdebug.profiler_output_dir = "<output folder name>"
```

3. Specify the name of the file to store snapshots in through the value of the `xdebug.profiler_output_name` directive. The default name is `cachegrind.out.%p`, where `%p` is the name format specifier. Accept the default name or define a custom one in compliance with the following standard:
   1. The name should always be `cachegrind.out`.
   2. Use the supported format specifiers .

Analyzing Xdebug Profiling Data

When integration with Xdebug profiler is enabled , PhpStorm provides visual representation of profiler snapshots. PhpStorm opens a separate editor tab with four views where the data are presented based on different criteria.

To have the profiling data collected and analyze it, perform these general steps:

- Initiating an Xdebug debugging session
- Retrieving the data accumulated by the profiler
- Examining the profiling data

Initiating an Xdebug debugging session

To start an Xdebug session, do one of the following:

- To start debugging an entire application, create debug configuration of the type PHP Web Application, and launch debugging by clicking the Debug toolbar button . See Debugging with a PHP Web Application Debug Configuration for details.
- To debug a specific PHP HTTP request, define a debug configuration of the type PHP HTTP Request, and launch debugging by clicking the Debug toolbar button . See Debugging a PHP HTTP Request for details.
- To initiate a zero-configuration debugging session:
  1. Toggle the Start Listen PHP Debug Connections button so it changes to . After that PhpStorm starts listening to the port of the debugging engine used in the current project . Ports for debuggers are set at the PhpStorm level in the Debug dialog box (File | Settings | PHP | Debug for Windows and Linux or PhpStorm | Preferences | PHP | Debug for OS X).
  2. Open the starting page of your application in the browser, choose the Start debugger bookmark to activate the debugging engine from the browser, re-load the current page (the starting page of the application), and then return to PhpStorm.

Retrieving the data accumulated by the profiler

1. On the main menu, choose Tools | Analyze Xdebug Profiler Snapshot.
2. In the Select Xdebug profiler snapshot dialog box, that opens, choose the folder and the file where the profiling data is stored .
   PhpStorm presents the collected profiling data in a separate editor tab with the name of the selected profiler output file.

Examining the profiling data

When you request on the accumulated profiling data, PhpStorm opens its visualized presentation in a separate editor tab. The tab is named after the selected profiler output file and consists of several views. Switch between the views to analyze the profiling data based on various criteria of analysis.

- In the Execution Statistics view, examine the summary information about execution metrics of every called function.
- In the Call Tree view, explore the execution paths of all called functions.
- To explore the execution paths of a specific function, select the function in question in the Call Tree view and view its callees in the Callees view.
- To explore all the paths that can result in calling a specific function, select the function in question in the Call Tree view and examine its possible callers in the Callers view.

Profiling with Zend Debugger

Besides interactive debugging , PhpStorm integration with Xdebug also supports profiling. PhpStorm provides visual representation of profiling data generated by Zend Debugger.

Before profiling with Zend Debugger, download, install and configure the components of the PHP development environment . Normally, these are a PHP engine, a web server, and the Zend Debugger tool.

Enabling Profiling with Zend Debugger

Xdebug profiler is incorporated in the Zend Debugger tool. Therefore you only need to download, install, and enable Xdebug itself.

To enable profiling with Zend Debugger, perform these general steps:

- Configuring Zend Debugger
- Configuring the way to toggle the profiler from the browser

Configuring Zend Debugger

1. Download and install the Zend Debugger tool.
2. Integrate Zend Debugger with the PHP engine .
3. Integrate Xdebug with PhpStorm .

Configuring the way to toggle the profiler from the browser

To specify the GET/POST or COOKIE parameters, do one of the following:

- Generate the bookmarklets to toggle the debugger through. These bookmarklets will appear on the toolbar of your browser. They provide control over the debugger cookie, through them you will activate and deactivate the debugger.
  1. Enable the Bookmarks toolbar in your browser by doing one of the following depending on the browser type:
     - In Firefox, choose View | Toolbar | Bookmarks Toolbar.
     - In Chrome, choose Bookmarks | Show bookmarks bar.
  2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Debug under PHP.
  3. On the Debug page, that opens, click the Use debugger bookmarklets to initiate debugger from your favorite browser link.
  4. On the Zend Debugger & Xdebug bookmarklets page that opens, check the debugging engine settings and click Generate. The bookmarks for listed debugging-related actions are generated.
  5. Drag the generated links to the bookmark toolbar in your browser.
- Specify the values manually.

Analyzing Zend Debugger Profiling Data

When integration with the Zend Debugger profiler is enabled , PhpStorm provides visual representation of profiler snapshots. PhpStorm opens a separate editor tab with four views where the data are presented based on different criteria.

With Zend Debugger, profiling is supported within a zero configuration debugging session .

To have the profiling data collected and analyze it, perform these general steps:

- Initiating a zero configuration Zend Debugger session
- Examining the profiling data

Initiating a zero configuration Zend Debugger session

1. Generate the bookmarklets to toggle the debugger through. These bookmarklets will appear on the toolbar of your browser. They provide control over the debugger cookie, through them you will activate and deactivate the debugger.
   1. Enable the Bookmarks toolbar in your browser by doing one of the following depending on the browser type:
      - In Firefox, choose View | Toolbar | Bookmarks Toolbar.
      - In Chrome, choose Bookmarks | Show bookmarks bar.
   2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Debug under PHP.
   3. On the Debug page, that opens, click the Use debugger bookmarklets to initiate debugger from your favorite browser link.
   4. On the Zend Debugger & Xdebug bookmarklets page that opens, check the debugging engine settings and click Generate. The bookmarks for listed debugging-related actions are generated.
   5. Drag the generated links to the bookmark toolbar in your browser.
2. Toggle the Start Listen PHP Debug Connections button so it changes to . After that PhpStorm starts listening to the port of the debugging engine used in the current project . Ports for debuggers are set at the PhpStorm level in the Debug dialog box (File | Settings | PHP | Debug for Windows and Linux or PhpStorm | Preferences | PHP | Debug for OS X).
3. Open the starting page of your application in the browser, choose the Start debugger bookmark to activate the debugging engine from the browser, re-load the current page (the starting page of the application), and then return to PhpStorm.
   Establishing a Zero-Configuration debugging session may fail, with no breakpoints hit and therefore the script not suspended. This may happen if if the path mappings are not configured or configured erroneously, or if you have not set any breakpoints. In the latter case, enabling selecting the Break at First Line in PHP Scripts check box in the External Connections area or turning the Run | Break at First Line in PHP Scripts option on the main menu may also help.

   To have PhpStorm display a notification if the script is not suspended, select the Notify if debug session was finished without being stopped check box in the Advanced Settings area on the Debug page of the Settings dialog box.
4. In the dialog box, that opens, select the incoming connection to profile and click Accept. The Incoming Connection from Zend Debugger dialog box appears only once, when you accept connection from this host for the first time.
   PhpStorm presents the collected profiling data in a separate editor tab with the name of the selected profiler output file.

Examining the profiling data

When you request on the accumulated profiling data, PhpStorm opens its visualized presentation in a separate editor tab. The tab is named after the file that implements the page you are currently profiling and consists of several views. Switch between the views to analyze the profiling data based on various criteria of analysis.

- In the Execution Statistics view, examine the summary information about execution metrics of every called function.
- In the Call Tree view, explore the execution paths of all called functions.
- To explore the execution paths of a specific function, select the function in question in the Call Tree view and view its callees in the Callees view.
- To explore all the paths that can result in calling a specific function, select the function in question in the Call Tree view and examine its possible callers in the Callers view.

Testing PHP Applications

The topics in this part provide guidelines in PHP-specific unit testing procedures. For general information on testing in PhpStorm, see the section Testing .

PhpStorm supports unit testing of PHP applications through integration with the PHPUnit tool.

Generally, PhpStorm runs and debugs PHPUnit tests same way as other applications, by running the run/debug configurations you have created . When doing so, it passes the specified test class, file, or directory to the test runner. You can run unit testing locally and remotely depending on the chosen run configuration.

To create and run unit tests on PHP applications, perform the following general steps:

- Enable PHPUnit support .
- Write the unit tests to run.
- Group the test to distinguish between testing in a production and in your development environment or to enable filtering tests by their authors.
- Create a run configuration :
  - To run unit tests locally, create a PHPUnit configuration.
  - To run unit tests on a remote server, create a PHPUnit by HTTP configuration.
- Launch unit tests by clicking the Run toolbar button and monitor test results in the Run tool window .
- Launch unit tests with coverage by clicking the Run with coverage toolbar button and analyze the test coverage in the Coverage Tool Window .
- Debug unit tests by setting breakpoints where necessary and clicking the Debug toolbar button . For details, see PHP Debugging Session .

Enabling PHPUnit Support

PhpStorm supports unit testing of PHP applications through integration with the PHPUnit tool.

You can use PHPUnit in PhpStorm in the following ways:

- Install PHPUnit and run it from PEAR that is configured as include path. The advantage of this approach is that once installed, PHPUnit can be re-used in other PhpStorm projects.
  The PHPUnit installation procedure depends on the operating system you use and your system settings. Please, refer to the PHPUnit installation instructions for information on installing and configuring this tool.
- Install PHPUnit using the Composer Dependency Manager and run this instance.
- Without installing PHPUnit. Just run it from the `phpunit.phar` archive.

On this page:

- Configuring the PEAR folder with PHPUnit as an include path
- Loading PHPUnit with autoload.php from the Composer dependency manager
- Running PHPUnit from the phpunit.phar archive

Configuring the PEAR folder with PHPUnit as an include path

The PHPUnit installation procedure depends on the operating system you use and your system settings. Please, refer to the PHPUnit installation instructions for information on installing and configuring this tool.

1. Open the Settings dialog box , and click PHP. The PHP page opens.
2. From the Interpreter drop-down list, choose the PHP installation to use.
3. Configure the `PEAR` folder as an include path . Click the Add button in the Include path area, and select the `PEAR` folder in the dialog box, that opens.
4. Click PHPUnit under the PHP node, and on the PHPUnit page that opens choose Load from include path.

Loading PHPUnit with autoload.php from the Composer dependency manager

The Composer Dependency Manager can download PHPUNit and add it to the current project. The tool loads packages using the `autoload.php` file in the `vendor` folder.

1. Enable Composer in your project. Do one of the following:
   - Download `composer.phar` , configure Composer as a command line tool , and initialize it in your project in the command line mode.
   - Set up Composer in your project through the user interface: choose Tools | Composer | Init Composer and then specify the location of an existing `composer.phar` file or have PhpStorm download the file. For more details, see Using Composer Dependency Manager .

     When the initialization process is completed, the `autoload.php` file is stored in the `vendor` folder.
2. Open the Settings dialog box , and click PHPUnit under the PHP node.
3. On the PHPUnit page that opens, choose Use custom loader and specify the location of the `autoload.php` in the Path to script text box below.

When you start a testing session for the first time, the `autoload.php` file starts, installs and runs PHPUnit.

Running PHPUnit from the phpunit.phar archive

You can download `phpunit.phar` manually and then tell PhpStorm to load the libraries from it or have PhpStorm download `phpunit.phar` automatically. Depending on your preferences, do one of the following:

- Download `phpunit.phar` manually:
  1. Download phpunit.phar manually.
  2. Open the Settings dialog box , and click PHPUnit under the PHP node.
  3. On the PHPUnit page that opens, choose Path to phpunit.phar and specify the location of the archive in the Path to phpunit.phar text box below.
- Have PhpStorm download `phpunit.phar` :
  1. Open the Settings dialog box , and click PHPUnit under the PHP node.
  2. On the PHPUnit page that opens, choose Path to phpunit.phar and click the Download phpunit.phar... link.
  3. In the dialog box that opens, specify the folder to store the `phpunit.phar` archive after download. PhpStorm starts the download. After the process is completed successfully the file is stored in the chosen folder and the path to this folder is displayed in the Path to phpunit.phar text box below.

Creating PHPUnit Tests

You can write unit tests manually or have test stubs generated automatically based on the PHP classes that are subject for testing.

A Test class is a PHP class with the name `<production class>Test.php`. For example, if the class to test is `Myclass.php`, the tests for it will be in `MyclassTest.php`.

PhpStorm also generates tests for classes defined among others within a PHP file. If several production classes are defined in one single file, for each generated test class PhpStorm will create a separate file.

Test classes are inherited from `PHPUnit_Framework_TestCase`.

The data for generating PHP test classes are specified in the Generate PHPUnit Test dialog box.

On this page:

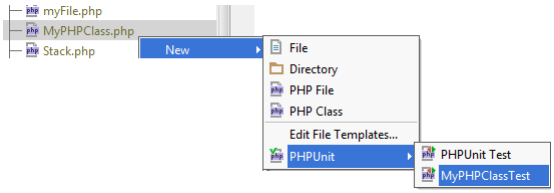
- Prerequisites
- Generating a test for a PHP class defined in a separate file
- Generating a test for a PHP class defined among others within a PHP file
- Generating a PHPUnit test method

Prerequisites

Make sure, the PHPUnit tool is installed on your machine and enabled in PhpStorm. For more information, see http://www.phpunit.de/manual/current/en/installation.html and Enabling PHPUnit Support.

Generating a test for a PHP class defined in a separate file

1. In the Project tool window, select the PHP class to create unit tests for, e.g. `MyPHPClass` as shown in the image below, and choose New | PHPUnit | PHPUnit Test on the context menu of the selection.



2. In the Generate PHPUnit Test dialog box that opens, specify the following data:
   1. The fully qualified name of the class to be tested in the Class to test area. The specified name will be used to propose the Test Class Name. To use completion, press Control+Space and choose the relevant production class from the pop-up list. By default, the Name text box displays the name of the class on which the test generation was invoked.
   2. The name of the test class. PhpStorm automatically composes the name from the production class name as follows: `<production class>Test.php`. The test class name is displayed in the Name text box of the Test Class area.
   3. The directory where the file with the test class will be stored. By default, it is the directory where the production class is stored. The default value is displayed in the Directory text box of the Test Class area. To specify another folder, click the Browse button and choose the relevant folder in the dialog box that opens. To use completion, press Control+Space and choose the relevant folder from the pop-up list.
   4. The namespace the test class will belong to. PhpStorm completes the namespace automatically based on the directory and displays the generated value in the Namespace text box. When the directory is changed, the namespace is changed accordingly. To use completion, press Control+Space and choose the relevant namespace from the pop-up list.
   5. The test file name. By default, the name is displayed in the Name text box and is the same as the test class name.

   Check, accept, or update the predefined settings and click OK to initiate the test generation.

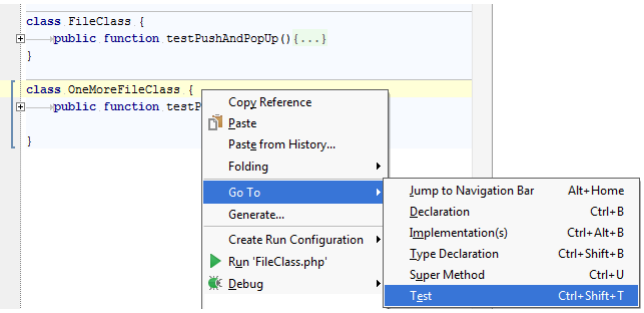
   Tip Alternatively, you select the PHP production class in question, choose New | PHPUnit | PHPUnit Test, and fill in the data manually.

When the test is ready, navigate back to the production class by choosing Navigate | Go to Test Subject. For details, see Navigating Between Test and Test Subject.

Generating a test for a PHP class defined among others within a PHP file

1. In the editor, open the file with the class to generate the test for.

2. Do one of the following:
   - On the main menu, choose File | New | PHPUnit | PHPUnit Test.
   - On the context menu, choose Go To | Test on the context menu, then choose Create New Test in the pop-up list.



3. In the Generate PHPUnit Test dialog box that opens, specify the following data:
   1. The fully qualified name of the class to be tested in the Class to test area. To use completion, press Control+Space and choose the relevant production

class from the pop-up list. The specified name will be used to propose the Test Class Name.

2. The name of the test class. PhpStorm automatically generates the name and displays it in the Name text box of the Test Class area.
3. The directory where the file with the test class will be stored. By default, it is the directory where the production class is stored. The default value is displayed in the Directory text box of the Test Class area. To specify another folder, click the Browse button and choose the relevant folder in the dialog box that opens. To use completion, press Control+Space and choose the relevant folder from the pop-up list.
4. The namespace the test class will belong to. PhpStorm completes the namespace automatically based on the directory and displays the generated value in the Namespace text box. When the directory is changed, the namespace is changed accordingly. To use completion, press Control+Space and choose the relevant namespace from the pop-up list.
5. The test file name. By default, the name is displayed in the Name text box and is the same as the test class name.

Check, accept, or update the predefined settings and click OK to initiate the test generation.

Generating a PHPUnit test method

Besides generating test class stubs, PhpStorm can help you populate them by generating stubs for test methods.

1. Open the required test class in the editor, position the cursor anywhere inside the class definition, and do one of the following:
   - On the main menu, choose Code | Generate, then choose PHPUnit Test Method from the Generate pop-up list.
   - On the context menu, choose Generate, then choose PHPUnit Test Method from the Generate pop-up list.

2. Set up the test fixture, that is, have PhpStorm generate stubs for the code that emulates the required environment before test start and returns the original environment after the test is over. For more details, see http://phpunit.de/manual/3.7/en/fixtures.html .
   1. On the context menu, choose Generate | Override method.
   2. From the Choose methods to override dialog box that opens, choose SetUp or TearDown and click OK.

Grouping Tests

Grouping tests is helpful, for example, to distinguish between tests to run in a production environment from those to run in your development environment. You just need to create two groups and then include or exclude them depending on the current environment.

In this section:

- Attaching a test to a group
- Filtering tests by their authors
- Including and excluding test groups from execution

To attach a test to a group

- Tag the test with the @group annotation in the format:

```
/**
@group <group specification>
*/
```

To enable filtering a test by its author

- Tag the test with the @author annotation in the format:

```
/**
@author <author specification>
*/
```

To configure execution of a test group

1. In the Run/Debug Configuration: PHPUnit by HTTP dialog box, open the Test groups tab.
2. To enable appointing groups to execute, select the Include/Exclude check box.
3. In the Group Name list, select the relevant group and do one of the following:
   - To have the tests in the selected group executed, select the Include check box.
   - To have the tests in the selected group skipped, select the Exclude check box.

Monitoring Code Coverage for PHP Applications

Code Coverage for PHPUnit shows you how much of your code is covered with tests and marks covered and uncovered lines visually right in the editor.

In this section:

– Prerequisites

– Configuring code coverage

– Measuring code coverage

Prerequisites

1. The PHPUnit tool is installed on your machine and enabled in PhpStorm .
2. A Xdebug or Zend Debugger debugging engine is configured in the current PHP interpreter.

To configure code coverage

1. Open the Settings dialog box .
2. Specify coverage options on the Coverage page.

To measure code coverage

1. Write the tests manually.
2. Create a run configuration of the type PHPUnit .
3. On the main toolbar, select the PHPUnit run configuration in the Run/Debug Configurations drop-down list and click the Run with Coverage button .
4. Monitor the code coverage in the Coverage tool window.

Using Behat Framework

With PhpStorm, you can practise behaviour-driven development by running scenarios using the BehatBehat framework. Currently PhpStorm supports integration with Behat 3 and Behat 2 versions.

Native support of Behat in PhpStorm includes:

- Recognition of and coding assistance for `.feature` scenario files and `.php` scenario definition files.
- Support of Gherkin syntax in `.feature` files: Feature, Scenario, Given, When, Then, And, and But keywords.
- Recognition of @given, @when, and @then annotations in definition files.
- Setting correspondence between scenarios and their definitions through regular expressions in accordance with the PCRE standard for Behat 2.4 and PCRE+ for Behat 3.0. Turnip expressions are also welcome.

On this page:

- Before you start
- Downloading and installing Behat
  - Downloading the behat.phar archive
  - Installing Behat using the Composer dependency manager
- Registering the Behat installation in PhpStorm
- Creating a run configuration of the type Behat
- Launching Tests

Before you start

1. Make sure the PHP interpreter is configured in PhpStorm on the PHP page , as described in Enabling PHP Support .
2. Make sure the PHP and Behat plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Configure the PHP interpreter to use Behat with, as described Configuring Remote PHP Interpreters . Note that Behat 3 requires PHP 5.5 and higher.

Downloading and installing Behat

You can acquire Behat in the following ways:

- Download the `behat.phar` archive.
- Use the Composer dependency manager.

Downloading the behat.phar archive

1. Download `behat.phar` at the Behat Downloads page .
2. Store `behat.phar` on your computer:
   - To get full coding assistance in addition to simply running Behat scenarios, store `behat.phar` under the root of the project where Behat will be later used. In this case, PhpStorm will include it in indexing, so references to Behat classes will be successfully resolved and you will be able not only to run scenarios but also provided with full coding assistance.
   - If you only need to run Behat scenarios and you do not need any coding assistance, you can save `behat.phar` outside the project. In this case, PhpStorm will only run Behat scenarios without providing coding assistance.

Installing Behat using the Composer dependency manager

Before you start, make sure Composer is installed on your machine and initialized in the current project, see Using Composer Dependency Manager .

1. On the main menu, choose Tools | Composer | Add Dependency.
2. In the Add Composer Dependency dialog that opens, select the `behat/behat` package from the Available Packages list, possibly using the search field. The list shows all the available packages, however, the packages that have been already installed are marked with a tick.
   Choose the relevant version from the Version to install drop-down list.
3. If necessary, expand the Settings hidden area and specify the advanced installation options. In the Command line parameters text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have the package added to the `require-dev` section instead of the default `require` section type `-dev` . For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .
4. Click Install.

When you click Create, the `create-project` command is invoked with the selected package. This results in creating a Composer project whose configuration and structure depends on the selected package, see https://getcomposer.org/doc/03-cli.md#create-project for details. After that a PhpStorm project opens.

Registering the Behat installation in PhpStorm

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Behat under PHP.
2. On the Behat that opens, specify the location of the Behat executable file or `behat.phar` archive and the PHP interpreter to use Behat with. This PHP interpreter can be installed either locally on your computer, or in a Vagrant virtual environment, or on a remote host. For information on configuring PHP interpreters, see Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters .
   Click the Add toolbar button + in the central panel and do one of the following depending on your workflow:
   - To use Behat with a local PHP interpreter:
     1. Choose Local.
     2. In the Behat Library area, specify the location of the Behat executable file or `behat.phar` archive.
     3. Click next to the Path to Behat directory or phar file text box. PhpStorm detects the version of Behat and displays it below the text box.
   - To use Behat in a Vagrant environment:
     1. Choose By Remote Interpreter. In the Behat by Remote Interpreter dialog box, choose the desired PHP interpreter configuration in a Vagrant environment. When you click OK, PhpStorm returns to the Behat page.
        PhpStorm displays the appointed PHP interpreter configuration in the Interpreter field. The field is read-only, to update the chosen interpreter configuration or create

a new one, click the ... whereupon PhpStorm brings you to the Interpreters dialog box .

2. In the Behat Library area, specify the location of the Behat executable file or `behat.phar` archive in the target Vagrant virtual environment. For example, if you installed Behat through the Composer dependency manager, the location is `vendor/bin/behat` .
3. Click next to the Path to Behat directory or phar file text box. PhpStorm detects the version of Behat and displays it below the text box.

– To use Behat on a remote host with SSH access:

1. Choose By Remote Interpreter. In the Behat by Remote Interpreter dialog box, choose the desired PHP interpreter configuration on a remote host. When you click OK, PhpStorm returns to the Behat page.
   PhpStorm displays the appointed PHP interpreter configuration in the Interpreter field. The field is read-only, to update the chosen interpreter configuration or create a new one, click the ... whereupon PhpStorm brings you to the Interpreters dialog box .
2. In the Behat Library area, specify the location of the Behat executable file or `behat.phar` archive on the target host.
3. Click next to the Path to Behat directory or phar file text box. PhpStorm detects the version of Behat and displays it below the text box.

3. In the Test Runner area, appoint the configuration `.yaml` file to use for launching and executing scenarios.
   By default, Behat looks for a `behat.yaml` configuration file in the project root folder or in the `config` folder. You can appoint a custom configuration file.

   – To have Behat use the `behat.yaml` configuration file from the project root folder or from the `config` folder. If no such file is found, test execution fails, therefore it may be more reliable to specify the configuration file explicitly. clear the Default configuration file check box.
   – To specify your own `.yaml` configuration file. Ths file will be later used as default in all Behat run/debug configurations.
   In the text box, specify the location of the configuration file to use. Type the path manually or click the ... button and choose the file in the dialog box that opens.

   select the Default configuration file check box.

Creating a run configuration of the type Behat

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button +, and choose Behat on the context menu.
3. In the Run/Debug Configuration: Behat dialog box , specify the scenarios to run and customize the behavior of the current PHP interpreter by specifying the options and arguments to be passed to the PHP executable file.

Launching Tests

For information about writing Behat features, see http://docs.behat.org/en/latest/guides/1.gherkin.html . To run or debug your tests, do one of the following:

– On the main menu, choose Run | Run <configuration name> and monitor test results in the Run tool window as described in Viewing and Exploring Test Results .

– On the main menu, choose Run | Debug <configuration name> and debug the tests in the Debug tool window .

PHP-Specific Command Line Tools

With PhpStorm, you can use PHP from the command line through integration with PHP-specific command line tools. Integration with command line tools is provided at the PhpStorm level, so once configured, a tool can be used in all your PhpStorm projects. Just activate or de-activate it when necessary depending on the needs of each specific project.

PhpStorm supports integration with third-party PHP-specific tools:

1. Symfony , version 1.1 or higher.
2. Symfony2 that incorporates the Twig template engine. PhpStorm supports the official Symfony2 code style for Twig and automatically inserts one space after an opening pair of curvy braces and before a closing pair of curvy braces in Twig templates:

```
{{ some_variable }}
```

3. Zend Framework .
4. Yii .

Alternatively, you can create your own tool and populate it with commands that fit your preferences and needs. The available tools are listed in the Command Line Tool Support page of the Settings dialog box.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

To work with a PHP command line tool, perform these general steps:

- Download and enable the required third party tool or define a custom tool .
- Create and open a Symfony or a Yii project, open it in PhpStorm, and monitor its structure in the Framework tool window.
- Customize the tool commands , if necessary, and have the tool definition file validated .
- Reload the commands from the executable file .
  This functionality is currently available for the Symfony tool only. You can add new tasks to the command definition file and enable PhpStorm to use them without re-configuring the command line tool, that is, without removing it from the list of available ones and adding it anew.
- Invoke commands and analyze the results of their execution in the dedicated Command Line Tools Console tool window.

Enabling a Command Line Tool

The PHP command line tools that are available from PhpStorm are listed in the Command Line Tool Support page of the Settings dialog box. Integration with command line tools is provided at the PhpStorm level, so once configured, a tool can be used in all your PhpStorm projects. Just activate or de-activate it when necessary depending on the needs of each specific project.

You can download and activate an existing third-party tool or create your own tool . You can update the command definitions of third-party or a custom tool right in the PhpStorm editor.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

On this page:

– Enabling integration with a third party PHP command line tool
– Creating a custom PHP command line tool

To enable integration with a third party PHP command line tool

Integration with command line tools is supported via the Command Line Tool Support plugin. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

1. Download and install the desired tool. Currently PhpStorm supports integration with the following tools:
   1. Symfony version 1.1 or higher.
   2. Zend Framework 1, see http://framework.zend.com/download/overview .
   3. Zend Framework 2 (ZFTool, see http://framework.zend.com/downloads/latest .
   4. Composer Dependency Manager .
   5. Drush , version 5.8 and higher.
   6. Symfony console-based tools Laravel and Doctrine . These tools are installed through Composer.

2. Open the PhpStorm settings and click Command Line Tool Support.

3. In the Command Line Tool Support page, click the Add button. In the Choose Tool to Add dialog box that opens, choose Zend Framework or Symfony.

4. In the dialog box that opens, specify the path to the definition file of the chosen tool in the Path to <tool> text box. Depending on the chosen tool, specify the location of the following file:
   – `<Zend-Framework-home>/zf.bat` for Zend Framework 1.
   – `<Zend-Framework-home>/zf.php` or `<Zend-Framework-home>/zf.php` for Zend Framework 2.
   – `<Symfony-home>/data/bin/symfony` for Symfony 1.*.
   – `<Symfony-home>/app/console` for Symfony 2.
   – `<Composer-home>/composer.phar` for Composer
   – `C:/ProgramData/Drush/drush.bat` on Windows and `/usr/bin/drush` on Mac OS or Linux for Drush.
   – For Laravel, the path to the artisan bundled with the tool, for example, `C:\MY_PROJECTS\PHP_PROJECTS\MyLaravelProject\artisan` .
   – For Doctrine, the path to the tool home.

   PhpStorm searches the contents of the specified file for definitions of commands. When the file analyzes is completed, PhpStorm displays the specified file in the list.

5. In the Alias text box, specify the alias to use in calls of tool commands. Accept the default alias or edit it, if necessary.

6. To activate the detected command set, select the Enable check box.

7. Customize the command set , if necessary.

8. In the Show console in area, specify where you want the Input pane for typing commands opened:
   – To have the Input pane opened in a pop-up window, choose the Pop-up option.
   – To have the Input pane opened as a text box at the bottom of the Command Line Tools Console tool window, choose the Tool window option.

To create a custom command line tool

1. Open the PhpStorm settings and click Command Line Tool Support.

2. In the Command Line Tool Support page, click +. In the Command Line Tools dialog box that opens, choose visibility level of the tool (project or global), and its type from the Choose tool drop-down list.

3. In the dialog box that opens, specify tool name and location, and the other required parameters.
   When you click OK, PhpStorm brings you to the Command Line Tool Support page, where the new tool is added to the list of available tools.

4. Select the newly created tool and click . In the tool definition `.xml` file that opens in the editor, define the tool commands.

5. In the Show console in area, specify where you want the Input pane for typing commands opened:
   – To have the Input pane opened in a pop-up window, choose the Pop-up option.
   – To have the Input pane opened as a text box at the bottom of the Command Line Tools Console tool window, choose the Tool window option.

Using Framework Model View Controller

With PhpStorm, you can examine the structure of a project in the terms of the Model-View-Controller pattern . This functionality is provided in the dedicated Framework tool window and is currently available for the Symfony2 and Yii command line tools.

In this section:

– Prerequisites

– Enabling the MVC view

– Examining the MVC structure of a tool-specific project

Prerequisites

1. The Command Line Tool Support and Framework MVC Structure Support plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
2. The Symfony2 or Yii command line tool is downloaded.
3. Integration with the tool is enabled on the Command Line Tool Support page of the Settings dialog box.
4. A Symfony2 or a Yii specific project is available on your machine.

To enable the MVC view

1. Open the PhpStorm settings and click Framework Integration.

2. In the Framework Integration page, specify the command line tool your project is based on and the location of the tool project. PhpStorm checks the contents of the specified tool project directory and accepts the appointed project root if it contains the following subfolders:
   – For Symfony2: `app` , `src` , `vendor` , and `web` subfolders.
   – For Yii: a `protected` subfolder.

After that, the Framework tool window is available through the View | Tool Windows menu.

To examine the MVC structure of a tool-specific project

1. To open the required tool-specific project, choose File | Open or File | Open Directory, and then select the project root folder in the dialog box that opens.

2. In the Framework tool window that immediately opens automatically, examine the project in the terms of bundles, controllers, and other elements of the MVC pattern.

Using Phing

PhpStorm supports integration with the Phing build framework. Phing is mainly intended for building PHP projects, but it can be also used as a build tool in a number of areas. Phing functionality in PhpStorm is provided through a dedicated Phing Build tool window.

In this section:

– Before you start

– Accessing Phing Buide tool window

Before you start

1. Make sure Phing is downloaded and set up on your computer. If you are using an AMP , the package may already contain Phing.
2. Make sure the Phing Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Accessing Phing Buide tool window

1. Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
2. Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

Enabling Phing Support

On this page:

- Introduction
- Before you start
- Enabling Phing support
  - Enabling Phing support through the project settings
  - Enabling Phing support from the Phing Build tool window
  - Enabling Phing on the fly, when attempting to run a build file

Introduction

At the project level, you can enable Phing in the following ways:

- Explicitly through the project settings, on the Phing page of the Settings/Preferences dialog box.
- When adding a file to the list of Phing build files. A correct Phing build file is an `xml` file with the root element `<project>`.
- On the fly, when you attempt to run a build from the editor or from the project tool window.

In either case, the Phing Build tool window becomes available from the View | Tool Windows | Phing Build menu within the scope of the current project.

Before you start

1. Make sure Phing is downloaded and set up on your computer. If you are using an AMP, the package may already contain Phing.
2. Make sure the Phing Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Enabling Phing support

To enable Phing support through the project settings

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Phing under PHP.
2. In the Phing page that opens, specify the location of the file `phing.bat` in the Path to Phing executable text box. Type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.

To enable Phing support from the Phing Build tool window

1. Open a Phing build file in the editor or select it in the Project tool window.

   Tip A correct Phing build file is an `xml` file with the root element `<project>`.

2. On the context menu of the editor or the selection, choose Add as Phing build file.
3. In the Phing Build tool window, that opens, click the Settings button on the toolbar.
   The button is only available if the list of build files is not empty.
4. In the Phing Settings dialog box, that opens, specify the location of the `phing.bat` executable file.

To enable Phing on the fly, when attempting to run a build file

1. Open an Phing build file in the editor.
2. On the context menu of the editor, choose Run Build File.
3. In the Phing Build tool window, that opens, click the Settings button on the toolbar.
   The button is only available if the list of build files is not empty.
4. In the Phing Settings dialog box, that opens, specify the location of the `phing.bat` executable file.

Creating a List of Phing Build Files

On this page:

– Introduction

– Accessing Phing Buide tool window

– Managing lists of build files

Introduction

A correct Phing build file is an `xml` file with the root element `<project>`. Build files are created outside the Phing Build tool window and are normally stored under your PHP project root folder.. Find more about writing Phing build files in Phing Getting Started .

Accessing Phing Buide tool window

1. Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
2. Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

Managing lists of build files

To configure a list of build files, perform these general steps

1. Open the Phing Build tool window.
2. To add a build file to the list, click the Add button + on the toolbar and choose the required `.xml` build file in the Select Phing Build File dialog box, that opens.
3. To remove a build file from the list, select the file and click the Remove button – on the toolbar.
4. To navigate to the source code of a build file, select the desired file and choose Jump to Source on the context menu of the selection.

Managing Phing Build Targets

Phing functionality in PhpStorm is provided through a dedicated Phing Build tool window.

In this section:

- Accessing Phing Buide tool window
- Examining the build targets defined in a build file
- Hiding targets
- Associating a shortcut with a Phing target
- Appointing targets for execution before running or debugging

Accessing Phing Buide tool window

1. Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
2. Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

Examining the build targets defined in a build file

1. Open the Phing Build tool window. The window shows all the build files as nodes.
2. Depending on your goal, do one of the following:
   - To view the build targets defined in a specific build file, expand the corresponding node by clicking next to the build file name. Note that this expanding does not affect the targets marked as hidden.
   - To fold or unfold the build targets defined in all build files, click respectively the Expand All button or the Collapse All button on the toolbar. Note, that unfolding does not affect the targets marked as hidden.
   - To navigate to the definition of a target in the source code, select the desired target and choose Jump to Source on the context menu of the selection.

Hiding targets

Among the targets defined in a build file, you may have some that are only called by other targets and are never run alone. You can suppress showing such targets in the build file tree by marking them as hidden.

Hidden targets do not become visible when you expand the node of a specific build file or click Expand All button .

You can mark a target as hidden both directly from the Phing Build tool window or from the Phing Settings dialog box. However, the hidden status can be removed only through the Phing Settings dialog box.

Depending on your goal, do one of the following:

- To mark one or several build targets as hidden, do one of the following:
  - In the Phing Build tool window, select the targets under their build file node and choose Mark to Hide on the context menu of the selection.
  - Select the build file in which they are defined and click the Settings button on the toolbar. Then in the Phing Settings dialog box that opens, switch to the Hiding targets tab and select the Hide check boxes next to the targets to be hidden.
- To remove the hidden status of a target:
  1. Select the build file in which it is defined and click the Settings button on the toolbar.
  2. In the Phing Settings dialog box that opens, switch to the Hiding targets tab where the Hide check boxes next to the names of the hidden targets are selected. Clear the Hide check boxes next to the targets for which you want the status hidden removed.

Associating a shortcut with a Phing target

You can associate a build target with a keyboard shortcut and execute commonly-used targets with a single key-stroke. If a Phing build file is added to the project, its targets appear under the Phing Targets node in the Keymap dialog box.

1. In the Phing Build tool window, right-click the desired build target.
2. On the context menu, choose Assign Shortcut.
3. In the Keymap dialog box that opens, configure the shortcut as described in Configuring Keyboard Shortcuts .

Appointing targets for execution before running or debugging

1. Open the Phing Build tool window.
2. Select the desired target and choose Before Run/Debug on the context menu of the selection.
3. In the Execute Target Before Run/Debug dialog box that opens, select the configurations before which you want the target executed.

Running Builds

Phing functionality in PhpStorm is provided through a dedicated Phing Build tool window.

In this section:

- Accessing the Phing Buide tool window
- Running builds
- Running separate build targets

Accessing the Phing Buide tool window

1. Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
2. Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

Running builds

1. Open the Phing Build tool window.
2. Select the required build file in the list and click the Run button on the toolbar or choose Run Build on the context menu of the selection.

Running separate build targets

Do one of the following:

- In the Phing Build tool window, select the required build target in the list and click the Run button on the toolbar or choose Run Target on the context menu of the selection.
- Use the shortcut associated with the target .

Customizing Build Execution by Configuring Properties Externally

A Phing build file normally contains property elements that can be configured either manually in the build file itself or externally, in the Phing Build tool window. Properties that are configured externally are passed to Phing through a command line when build execution is launched and therefore they are every time re-calculated dynamically. Externally configured properties always override the ones set manually in the build file.

On this page:

- Accessing Phing Buide tool window
- Configuring a property externally

Accessing Phing Buide tool window

1. Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
2. Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

Configuring a property externally

1. Open the Phing Build tool window.
2. Select the required build file in the list and click the Settings button on the toolbar.
3. In the Phing Settings dialog box that opens, switch to the Properties tab and configure a list or properties to be passed to Phing through a command line.
   1. To add a new property to the list, click the Add button.
   2. In the Property text box, type the name of the property.
   3. In the Value text box, specify the property value to be passed. Do one of the following:
      - Type the value manually.
      - To use PhpStorm macros , click the Insert macro button + and configure a list of relevant macro definitions in the Macros dialog box that opens. To add a macro, select it in the list and click Insert.
4. When the list of properties is ready, click OK in the Phing Settings dialog box.

Using Drupal with PhpStorm

You can use PhpStorm as an IDE for Drupal development including modules, themes, and core. The supported versions are 6, 7, and 8 .

PhpStorm provides integration between the Symfony2 framework and Drupal 8 while developing Drupal modules and core.

In this part:

- Preparing for Drupal Development in PhpStorm
- Drupal-Specific Coding Assistance

Preparing for Drupal Development in PhpStorm

On this page:

- Before you start
- Creating a PhpStorm project with a Drupal module
- Importing a Drupal project
- Configuring the Drupal-specific development environment
  - Configuring file associations
  - Changing the Drupal settings

Before you start

1. Download and install Drupal .
2. Make sure the Drupal plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Creating a PhpStorm project with a Drupal module

You can create a PhpStorm project that will be set up and configured in accordance with the Drupal requirements. This is done by creating a project by a Drupal Module template.

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Drupal Module.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Drupal-specific information:
   1. Specify the root folder of the Drupal installation in the Drupal installation path. Type the path manually or click the Browse button and select the relevant folder in the dialog box that opens.
   2. Select the Set up PHP | Include paths check box to have Drupal include paths automatically configured for the project. The following paths will be added to the Include Paths list on the PHP page of the Settings dialog box:
      1. `<drupal installation root>/includes`
      2. `<drupal installation root>/modules`
      3. `<drupal installation root>/sites/all/modules`

      The include paths are added to the list after you leave the dialog box.
   3. From the Version drop-down list, choose the version of Drupal to use, the supported versions are 6, 7, and 8 .

You can update integration settings at any time later in the File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.

PhpStorm generates a project stub in accordance with the selected Drupal version. For Drupal 8, a `module_name.info.yml` file is generated where information about the module is provided.

Importing a Drupal project

Besides creating a project anew according to the Drupal Module template, you can open an existing Drupal project in PhpStorm. PhpStorm recognizes the Drupal-specific structure and suggests activating the Drupal support in the project.

1. Open the folder with the required Drupal project by choosing File | Open Directory.
2. As soon as PhpStorm detects the Drupal-specific structure in the project, the following message is displayed in a pop-up window:

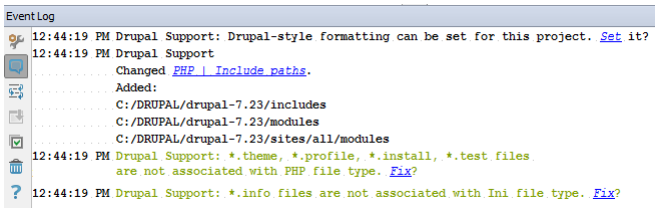
   Click the Enable link.
3. In the Drupal Module dialog box that opens, select the Enable Drupal integration check box and proceed as when creating a project with a Drupal module:
   1. Specify the root folder of the Drupal installation in the Drupal installation path.
   2. Select the Set up PHP | Include paths check box to have Drupal include paths automatically configured for the project.
   3. From the Version drop-down list, choose the version of Drupal to use, the supported versions are 6, 7, and 8 .

You can update integration settings at any time later in the File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.

Configuring the Drupal-specific development environment

Whether you enable the Drupal support in an existing PhpStorm project or create a new project with a Drupal module, PhpStorm checks if the development environment is configured properly for Drupal development.

Any detected inconsistency is reported in the Event Log tool window and as a pop-up. For each discrepancy PhpStorm suggests a fix. To apply a suggestion, click the link next to the reported event.



Configuring file associations

To enable code highlighting, you need to set some additional file type associations to make PhpStorm recognize and treat them as `php` files. Do one of the following:

- In the Event Log tool window, click the Fix link next to the `Drupal support: <*.file extension> files are not associated with PHP file type` message.
- Use the File Types page of the Settings dialog box:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and

click File Types under Editor.

2. In the Registered Patterns area, add the patterns that defined Drupal files to be recognized as `php` files. To do that, click the Add button + and type the pattern in the Add Wildcard dialog box that opens.

Changing the Drupal settings

No matter whether you enable the Drupal support in an existing PhpStorm project or create a new Drupal module, you can update the Drupal settings at any time on the Drupal page of the Settings / Preferences Dialog .

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Drupal under Languages&Frameworks.
2. Click Drupal and make the necessary changes on the Drupal page that opens:
   - To turn off the Drupal integration, clear the Enable Drupal integration check box. Note that this automatically excludes the folders specified as include paths.
   - Change the Drupal installation directory and the version of Drupal to use.
3. To change the include paths, click PHP and make the required changes in the Include Paths area of the PHP page that opens. Note that is the Drupal integration for is disabled, the paths are automatically excluded.
4. To change the code style settings, click Code Style, and then click PHP. On the Code Style: PHP page that opens, update the settings as necessary.

Drupal-Specific Coding Assistance

On this page:

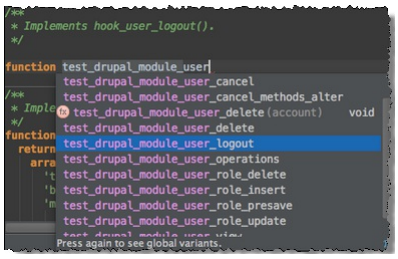
- Using Drupal hooks in PhpStorm
  - Completing hook declarations
  - Navigating to a hook invocation from the editor
  - Quick Documentation look-up for hooks
- Setting the Drupal code style in a project
- Checking code against the Drupal coding standards
- Viewing Drupal API documentation from PhpStorm
- Using the Drush command line tool
- Integration between Drupal 8 and Symfony2
  - Coding Assistance for Drupal8 - Symfony2 Integration

Using Drupal hooks in PhpStorm

PhpStorm provides full native support of Drupal hooks in `.module` files.

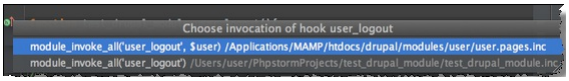
Completing hook declarations

PhpStorm indexes any hook invocation whereupon hook names become available in code completion for creating hook implementations. Start typing a hook name and press `Ctrl+Space`.



Navigating to a hook invocation from the editor

- To navigate to a hook invocation from the editor, click the icon in the gutter.
- In case of multiple invocations, PhpStorm displays a list of available hook invocations so you can choose which one to navigate to.



You will be navigated to the line where the relevant hook is invoked with `module_invoke_all()`, `module_invoke()`, or `drupal_alter()`.

Quick Documentation look-up for hooks

You can view quick documentation for a Drupal hook in PhpStorm. Documentation is taken from `.api.php` files provided by `core` and many other modules for reference purposes.

To view documentation on a hook, position the cursor at the hook name in question and choose View | Quick Documentation Lookup or press `Ctrl+Q`.

Setting the Drupal code style in a project

PhpStorm automatically offers to apply the pre-configured Drupal Coding Standards (code style) if a project is recognized as a Drupal Module, or if the Drupal integration is enabled in an existing project, or when you create a new project with a a Drupal Module. However, you can at any time change or customize this setting on the Code Style: PHP page of the Settings dialog box.

To have the pre-configured Drupal code style applied to a project, do one of the following:

- In the Event Log tool window, click the Set link next to the `Drupal-style formatting can be set for this project` message.
- Use the Code Style: PHP page of the Settings dialog box:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click PHP under Code Style.
  2. On the Code Style: PHP page that opens, click the Set from link.
  3. On the pop-up menu, choose Predefined, then choose Drupal.

Checking code against the Drupal coding standards

With PhpStorm, you can use two tools that detect violations against the Drupal coding standard: Coder and PHP Code Sniffer. They ensure your code remains clean and consistent and help prevent some common semantic errors made by developers.

- To use PHP Code Sniffer:
  1. Install PHP Code Sniffer as a PEAR package or using the Composer tool. For detailed installation instructions, see https://github.com/squizlabs/PHP_CodeSniffer.
  2. Register PHP Code Sniffer with PhpStorm and configure it as a PhpStorm inspection. For details, see Using PHP Code Sniffer Tool.
- To use Coder, the native Drupal code quality tool:
  1. Download the Drupal Coder module (7x-2.0 version is recommended). You don't need to install or use the Drupal Coder module, it contains the Drupal Coding Standards inside.
  2. Unpack the downloaded archive and find the `coder_sniffer/Drupal` subdirectory inside. You need to move the contents of this directory to `<php installation folder>/CodeSniffer/Standards/Drupal`. This directory must contain the `ruleset.xml` file and some other subdirectories.

Viewing Drupal API documentation from PhpStorm

To reach the Drupal API Documentation at https://api.drupal.org/api/drupal right from PhpStorm, select the symbol you are interested in and choose Search in Drupal API on the context menu of the selection.

Using the Drush command line tool

Drush is a command line shell and scripting interface for Drupal. With PhpStorm, you can use Drush 5.8 and higher.

1. Download and install Drush as described at https://github.com/drush-ops/drush .
2. Integrate Drush with PhpStorm by configuring it a s Command Line Tool:
   1. Open the Settings dialog box and click Command Line Tool Support.
   2. On the Command Line Tool Support page, click the Add button. In the Choose Tool to Add dialog box that opens, choose Drush and specify the tool's visibility. The available options are Project (available only in the current project) and Global (available across all PhpStorm projects). Click OK.
   3. In the Drush dialog bo that opes, specify the path to the Drush executable file. PhpStorm automatically fills in the default location, which is usually `C:/ProgramData/Drush/drush.bat` on Windows and `/usr/bin/drush` on Mac OS or Linux. If you followed the standard installation procedure, the predefined path will be correct, just click OK, whereupon PhpStorm loads command definitions automatically and returns to the Command Line Tool Support page.
   4. In the Alias text box, specify the alias to use in calls of tool commands. Accept the default alias or edit it, if necessary.
   5. To activate the detected command set, select the Enable check box.
3. To run a Drush command, choose Tools | Run Command... on the main menu or press `Ctrl+Shift+X` (`Cmd-Shift-X` on OS X).Ctrl+Shift+X (Cmd-Shift-X on OS X).

For details about configuring and running command line tools in PhpStorm, see Using Command Line Tools .

Integration between Drupal 8 and Symfony2

PhpStorm provides close integration between Drupal, version 8 and Symfony2 which connects Symfony2 components and Drupal infrastructure. To take advantage from this integration:

1. Install and activate the Drupal Symfony2 Bridge plugin on the Plugins page of the Settings dialog box as described in Installing, Updating and Uninstalling Repository Plugins and in Enabling and Disabling Plugins .
   The Drupal Symfony2 Bridge plugin depends on the Symfony2, which will be installed automatically.
2. To get advanced annotations support, install the PHP Annotations plugin as described in Installing, Updating and Uninstalling Repository Plugins and in Enabling and Disabling Plugins .

Coding Assistance for Drupal8 - Symfony2 Integration

- Completion for major Drupal-specific parameters in `<module_name>.info.yml` files that contain information about Drupal modules.
- Strings suitable for use inside Drupal-specific t() function are indexed across your project and offered for completion.
- Completion for relevant `.yml` key values in url() and other Drupal API functions, which makes search for the right value easier.
- Navigation to the `.yml` file by pressing `Ctrl+B` or choosing Navigate | Go To Declaration.
- Full support for service containers described in `.yml` files, including completion and navigation with `Ctrl+B`.
- Support of the Twig template engine , which is now the default template engine for Drupal 8, including completion, navigation, and recognition of Drupal functions. See also Twig in Drupal 8 .

Using Joomla! with PhpStorm

Use PhpStorm as an IDE for Joomla! development.

In this part:

- Using Joomla! with PhpStorm
  - Before you start
  - Joomla! support
- Creating and Importing Joomla! Projects
- Configuring Joomla! Support

Before you start

1. Download and install Joomla! .
2. Before you start working with Joomla!, make sure that the Joomla! Integration plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Joomla! support

Joomla! support includes:

- Ability to create a new project .
- `JHTML::_($argument)` , `JText::_($argument)` , `JText::script()` , `JText::sprintf();` support.
  When `JHTML::_($argument)` is used, PhpStorm navigates with `Ctrl+Click` on the first argument to the corresponding class or method.

  For example, consider the following code:

```
<?php
$options[] = JHtml::_('select.option', $eid, $extension_name);
```

  `Ctrl+Click` on `select.option` leads to navigation to the `option` method of the class `select` .

  When all the other constructs are used, PhpStorm navigates with `Ctrl+Click` on the first argument to the corresponding property in the `*.ini` file. For example, in the code

```
<?php
$msg = JText::sprintf('COM_INSTALLER_INSTALL_ERROR', JText::_('COM_INSTALLER_TYPE_TYPE_'
. strtoupper($package['type'])));
```

  `Ctrl+Click` on `'COM_INSTALLER_INSTALL_ERROR'` leads to navigation to the property `'COM_INSTALLER_INSTALL_ERROR'` in the file `en-GB.com_installer.ini` .
- Joomla! code style can be selected for the code in the PHP page of the Editor settings, when clicking the link Set from .
- PhpStorm detects Joomla when opening a Joomla! module/plugin/extension or a Joomla! root folder, and suggests enabling Joomla! support, and adjusting namespaces.
- DocBlocks standards for PHP files, classes, class properties, etc. When Joomla! support is recognized, PhpStorm suggests installing DocBlocks:

- PhpStorm suggests importing the Joomla! code style. See section Configuring Joomla! Support
- PhpStorm detects databases in projects. Just click + in the Database Tool Window tool window and choose Import from sources....

  The settings specified in the file `configuration.php` are detected and used for the new data source connection.
- PhpStorm provides database prefixes support and changes `#__` to the prefix that is defined in the `$dbprefix` field in the `configuration.php` file.
  It is worth noting that a database dialect should be selected in the SQL Dialects page of the Settings/Preferences dialog. Note also that the type of the selected dialect should be equal to your database type.

Creating and Importing Joomla! Projects

On this page:

- Creating a PhpStorm project with Joomla! Intergation support
- Importing a Joomla! project

Creating a PhpStorm project with Joomla! Intergation support

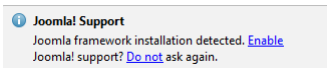
You can create a PhpStorm project that will be set up and configured in accordance with the Joomla! requirements. This is done by creating a project by a Joomla! Integration template.

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Joomla! Integration.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Joomla!-specific information:
   1. Specify the root folder of the Joomla! installation in the Joomla! installation path. Type the path manually or click the Browse button ... and select the relevant folder in the dialog box that opens.
   2. Select the desired Joomla! project type (component, module, or plugin).

Importing a Joomla! project

Besides creating a project anew according to the Joomla! template, you can open an existing Joomla! project in PhpStorm. PhpStorm recognizes the Joomla!-specific structure and suggests activating the Joomla! support in project.

1. Open the folder with the required Joomla! project by choosing File | Open Directory.
2. As soon as PhpStorm detects the Joomla!-specific structure in the project, the following message is displayed in a pop-up window:



   Click the Enable link.
3. In a message about the detected PSR roots, choose to configure the roots automatically.
4. In a message about DocBlock templates, click the link Install.

Note the following:

- You can update integration settings at any time later in the File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.
- When using the Joomla! Support pop-up:
  - If you prefer NOT enabling Joomla! support, just ignore the pop-up, and it will vanish.
  - The decision "Do not ask again" is project-wide.
  - You can always enable Joomla! support as described in the section Configuring Joomla! Support .

Configuring Joomla! Support

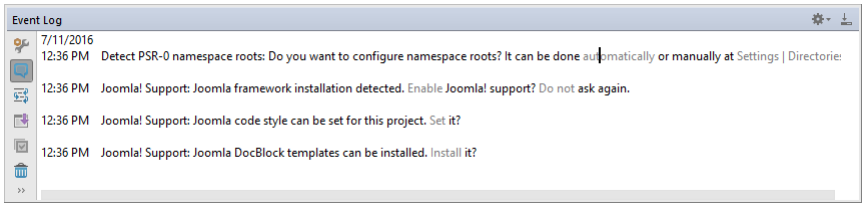
On this page:

- Configuring the Joomla!-specific development environment
- Changing the Joomla! settings

Configuring the Joomla!-specific development environment

PhpStorm checks if the development environment is configured properly for Joomla! development.

Any detected inconsistency is reported in the Event Log tool window and as a pop-up. For each discrepancy PhpStorm suggests a fix. To apply a suggestion, click the link next to the reported event.



Changing the Joomla! settings

You can update the Joomla! settings at any time on the Joomla! Support page.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Joomla! Support under Languages and Frameworks.
2. Make the necessary changes on the Joomla! Support page that opens:
   - To turn off the Joomla! integration, clear the Enable Joomla! integration check box.
   - Change the Joomla! installation directory.
3. If you did not set code style in the Events Log, then in the Settings/Preferences dialog, click Code Style, and then PHP. On the Code Style: PHP page that opens, update the settings as necessary.

Using Composer Dependency Manager

On this page:

- Introduction
- Preparing to run Composer in the command line mode
- Using Composer via the dedicated user interface
  - Setting up Composer in an existing project
  - Quick Composer setup
  - Adding Composer dependencies
  - Generating a Composer project stub

Introduction

The Composer Dependency Manager is very helpful when you need to install PHP-related frameworks, for example PHPUnit or Behat test frameworks.

You can use Composer from PhpStorm in three modes:

- Using the dedicated PhpStorm interface for most basic operations such as initializing a project with a `composer.json` stub created and adding dependencies .
- To have all Composer commands at disposal, you need to configure Composer as a command line tool and use it in the command line mode from the dedicated Command Line Tools Console Tool Window or pop-up window. For more details, see Running Command Line Tool Commands .
- You can have PhpStorm generate a project stub from an existing package using Composer. In this case, Composer will be initialized and one package will be installed during the project stub creation. After that you can use Composer from the command line or through the user interface.

Preparing to run Composer in the command line mode

You can use the full range of Composer commands from PhpStorm only in the command line mode. To use Composer in this mode, you need to configure it as a command line tool. For more details, see Using Command Line Tools .

1. Download the Composer.phar definition file .
2. Make sure the Command Line Tool Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Command Line Tool Support under Tools.
4. On the Command Line Tool Support page, click the Add button. In the Choose Tool to Add dialog box that opens, choose Composer.
5. In the Composer dialog box that opens, choose the way to launch Composer:
   - Composer.phar or php script: Choose this option to launch Composer through a PHP script or have PhpStorm detect and start the launcher in the `composer.phar` archive. In this mode, PhpStorm provides coding assistance and allows to execute scripts.
     In the Path to PHP executable field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.

     In the Path to composer.phar or composer text box, specify the location of the `composer.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
   - Composer executable: choose this option to launch Composer through the `Composer` executable file. In this mode, you do not get coding assistance and cannot execute scripts because not PHP engine is appointed for it.
     In the Path to executable text box, specify the location of the `composer` executable file. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.

   PhpStorm parses the contents of the specified `.phar` archive or executable file for Composer commands. When the file analyses is completed, PhpStorm returns to the Command Line Tools page of the Settings dialog bos where the specified file is added to the list of command line tools available in PhpStorm.

   You can have several instances of Composer configured and switch between them from one project to another by specifying the relevant one during Composer initialization .
6. In the Alias text box, specify the alias to use in calls of tool commands when running in the command line mode. Accept the default alias `c` or edit it, if necessary.
7. To activate the detected command set, select the Enable check box.
8. Customize the command set , if necessary.
9. In the Show console in area, specify where you want the Input pane for typing commands opened:
   - To have the Input pane opened in a pop-up window, choose the Pop-up option.
   - To have the Input pane opened as a text box at the bottom of the Command Line Tools Console tool window, choose the Tool window option.

For information about running the tool in the command line mode, see Running Command Line Tool Commands .

Using Composer via the dedicated user interface

PhpStorm provides a dedicated interface for such basic Composer operations as initializing a project and adding dependencies . In the UI mode, Composer is available at the project level. For information on using the tool from the command line, see Running Command Line Tool Commands .

Setting up Composer in an existing project

The dependencies added to a project through Composer are listed in the project-specific `composer.json` file. Therefore setting up the tool in a project starts with creating a project-specific `composer.json` file. This operation is also referred to as Composer initialization.

1. Open the project to use Composer.
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Composer under PHP.
3. On the Composer page that opens, specify the following:
   1. In the Path to PHP executable field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.
   2. In the Path to composer.phar text box, specify the location of the `composer.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.

3. If you have not downloaded the `composer.phar` file yet, click the Download link to have PhpStorm download `composer.phar` from the official storage and specify the folder to store the archive in. This instance of Composer will be available in the current project only. To use it in the command line mode, configure it as a command line tool .
4. Specify how you want PhpStorm to treat Composer packages:
   - To have PhpStorm mark Composer packages added to the current project as libraries., select the Add packages as libraries check box.
     In the context of PhpStorm, any file marked as library is automatically write-protected. Library files are indexed for the only purpose which is enhancing coding assistance (i.e. code completion, syntax highlighting, navigation, and documentation lookup). Library files are not to be edited in the scope of the project where they are declared as library.

     By default, the Add packages as libraries check box is selected, which automatically blocks the possibility to edit them in the current project.
   - If you want to use Composer packages in your development and at the same time update them yourself within the scope of the same project, clear the Add packages as libraries check box.

4. Click OK to save the Composer settings and leave the Settings dialog box.
5. On the main menu, choose Tools | Composer | Init Composer. PhpStorm creates a stub of the `composer.json` file and opens it in the editor.
6. Complete the code or accept the default values. For more details, see Composer.json: Project Setup .

Quick Composer setup

Alternatively, you can specify the location of the `composer.phar` archive and the PHP engine right during the Composer initialization:

1. On the main menu, choose Tools | Composer | Init Composer.
2. In the Composer Settings dialog that opens, specify the location of the `composer.phar` archive and the PHP engine to use. If you have not downloaded the `composer.phar` file yet, click the Download link to have PhpStorm download it now and specify the folder to store the file in.
3. Click OK. The dialog box is closed whereupon PhpStorm creates a stub of the `composer.json` file and opens it in the editor.
4. Complete the code or accept the default values. For more details, see Composer.json: Project Setup .

Adding Composer dependencies

When you add a dependency to the project using Composer, a new subfolder is created under the Vendor folder, the added package is stored in this subfolder, and the added package is added to the list in the `require` section of `Composer.json`. For information on adding dependencies manually in the command line mode, see Running Command Line Tool Commands . Below are instructions for adding dependencies using the interface provided by PhpStorm.

1. Open the project where Composer is set up and choose Tools | Composer | Add Dependency on the main menu.
2. In the Add Composer Dependency dialog that opens, select the packageToAdd from the Available Packages list, possibly using the search field. The list shows all the available packages, however, the packages that have been already installed are marked with a tick.
   Choose the relevant version from the Version to install drop-down list.
3. If necessary, expand the Settings hidden area and specify the advanced installation options. In the Command line parameters text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have a dependency added to the `require-dev` section instead of the default `require` section type `-dev`. For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .
4. Click Install.

When you click Create, the `create-project` command is invoked with the selected package. This results in creating a Composer project whose configuration and structure depends on the selected package, see https://getcomposer.org/doc/03-cli.md#create-project for details. After that a PhpStorm project opens.

Generating a Composer project stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Composer Project.
3. In the right-hand pane , in the Location field, specify the folder where the project will be created and provide the following Composer-specific information:
   1. Appoint the Composer instance to use.
      - To use commands from a previously downloaded `composer.phar`, choose Use existing composer.phar: and specify it location in the text box.
      - To have a new instance of Composer downloaded, choose Download composer.phar from getcomposer.org:. The `composer.phar` file will be saved under the project root folder specified in the Location text box of the New Project Dialog dialog.
   2. In the Package area, specify the package to install during the project creation. Select the package to add from the Available Packages list, possibly using the search field, and choose the relevant version from the Version to install drop-down list.
   3. Expand the Settings area and specify the path to the PHP interpreter to use (see Configuring Remote PHP Interpreters and Configuring Local PHP Interpreters for details). In the Command line parameters text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have a package added to the `require-dev` section instead of the default `require` section type `-dev`. For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .

When you click Create, the `create-project` command is invoked with the selected package. This results in creating a Composer project whose configuration and structure depends on the selected package, see https://getcomposer.org/doc/03-cli.md#create-project for details. After that a PhpStorm project opens.

Working with Diagrams

In this part:

- Working with Diagrams
  - Prerequisites
  - Basics
  - Features
- Viewing Diagram
- Adding Node Elements to Diagram
- Creating Node Elements and Members
- Creating Relationship Links Between Elements
- Deleting Node Elements from Diagram
- Navigating to Source from Diagram
- Viewing Changes as Diagram
- Viewing Class Hierarchy as a Class Diagram
- Viewing Siblings and Children
- Viewing Ancestors, Descendants, and Usages

Prerequisites

Before you start working with UML diagrams, make sure that the UML Support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Basics

UML model in PhpStorm is represented by a Class diagram in standard notation.

PhpStorm enables using UML class diagrams to analyze PHP applications, and the structure of the databases and tables. Besides that, you can explore changes committed to VCS.

Features

In PhpStorm, Class diagram features:

- Ability to view UML model as a diagram in a separate editor tab, in a pop-up window , or as a preview .
- Ability to invoke UML class diagram from the Project, Structure, Database, Version Control tool windows, the History tab of the Version Control tool window, and Navigation bar.
  In the editor, one can view class diagram for the whole class.
- Navigation from a diagram element to the underlying source code .
- Highlighting siblings and children classes .
- Refactorings (Rename , Move , Safe Delete , Inline).
- Navigation to class, file or symbol by name and to the last edit location .
- Viewing class information at the tooltip, and quick documentation lookup .
- Viewing changed classes as a UML Class diagram.
- Quick hierarchy view in a UML Class diagram pop-up window .
- Viewing subtypes and super classes .
- Ability to find usages of a node element or member.

Viewing Diagram

On this page:

- Prerequisites
- Basics
- Opening a UML class diagram
- Tips and tricks

Prerequisites

Before you start working with UML diagrams, make sure that the UML Support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Basics

You can invoke UML class diagram from different places:

- From the various tool windows.
- From the Navigation bar.
- From the Structure tool window.

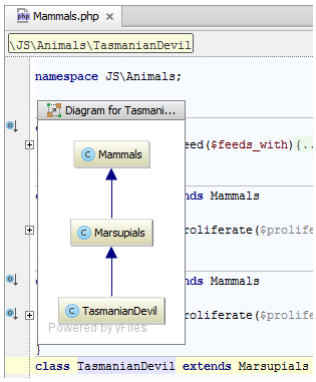
PhpStorm displays UML diagrams in two modes:

- In a pop-up window.
- In a separate editor tab.

Opening a UML class diagram

1. Select the desired item.
2. Do one of the following:
   - On the context menu of the selection, click Diagram, and on the submenu, select the way you want to view the model: Show Diagram or Show Diagram Pop-up.
   - Press `Ctrl+Shift+Alt+U` or `Ctrl+Alt+U`.

The diagram is displayed in the editor tab or in the pop-up window:



Tips and tricks

- You can open a UML class diagram without using your pointing device. Consider such a workflow: press `Alt+Home`, then press `Ctrl+Alt+U`.

Adding Node Elements to Diagram

You can add the existing node elements to the background of the UML Class diagram, using the context menu action, or drag-and-drop technique.

If there are relationships between the node elements in model, and the added element, these relationships will be displayed in diagram.

On this page:

- Adding node elements
- Dragging node elements
- Adding notes

To add a node element to a UML Class diagram

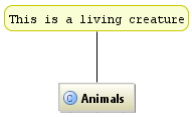
1. In the Enter class name to add dialog box, start typing the desired class name. PhpStorm automatically displays suggestion list with matching names. Note that you can include classes outside the scope of your project, by selecting the Include non-project classes check box.
2. Select the desired class from the suggestion list, and press `Enter`.

To add a node element using drag-and-drop technique

1. Select one or more desired elements in the Project tool window.
2. Drag selection to the diagram background.

To add a note to a node element

1. Right-click a node element, which you would like to comment.
2. On the context menu, choose New | Note.
3. In the text box, type the desired text. Note that `Enter` starts a new line in the text box. To complete a note, click OK, or press `Ctrl+Enter`.



Creating Node Elements and Members

You can populate your model with node elements and delete them from diagram, or from your project. When a node element is created in a UML Class diagram, the corresponding stub file is created in the current package.

In a similar way, PhpStorm helps create members in the node elements.

It is also possible to clarify the contents of diagrams with notes, which are created same way.

To create a node element in a UML Class diagram

1. Do one of the following:
   - Right-click the diagram background. Next, on the context menu, point to New, and choose element type from the submenu.
   - Press `Alt+Insert`, and choose element type from the pop-up frame.

2. Specify the element name, or type the contents of a note, and click OK.

To create a member in a node element

1. Select a node element in a diagram.

2. Do one of the following:
   - On the context menu of the selection, point to New, and then choose the member type on the submenu.
   - Press `Alt+Insert`, and choose member type from the pop-up frame.

3. Depending on the selected member type, specify the member name and the other parameters. See preliminary results in the Preview pane of each dialog.

Creating Relationship Links Between Elements

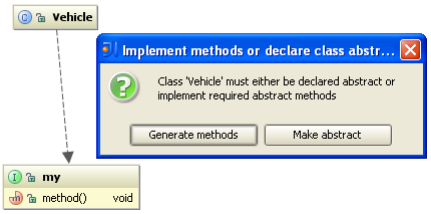
For creating relationship links between node elements, PhpStorm provides a special mode, in which drag-and-drop technique can be used for drawing links. As a link is created, the corresponding extends / implements clause is generated in the underlying source code. So doing, the extends link is represented with a solid line, and the implements link is represented with a dotted line.

To create a link between node elements

1. Make sure that is pressed on the diagram toolbar.

2. Draw a link from the source to a target node.

3. Specify whether the target class should be declared abstract, or implement required abstract methods:



To delete a link

1. Select a link between two node.

2. Press `Delete`.

3. In the dialog box that opens, confirm deletion:

To remove elements from view

1. In the diagram, select one or more elements to be deleted.
2. Press `Delete`.

Navigating to Source from Diagram

In a diagram, you can use all the regular procedures that enable navigating to the underlying source code:

- Jump to source (`F4`). So doing, if the Jump to Source command has been invoked on a class node, the caret is placed at the class declaration.
- View source (`Ctrl+Enter`)
- Navigate by name (`Ctrl+N` , `Ctrl+Shift+N` , or `Ctrl+Shift+Alt+N`)

Viewing Changes as Diagram

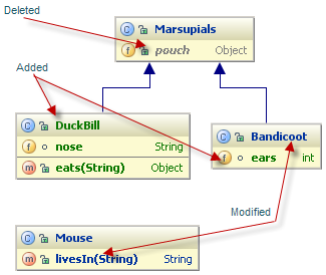
For the modules under version control, you can usually view the list of changed files in the Changes tool window. If you want to evaluate how your changes affect the model, use UML Class diagram. This view presents the complete picture of changes, including relationships between the modified classes.

For this purpose, PhpStorm provides the action Show changed classes , which is available from the editor, Project tool window, and the Local Changes tab of the Version Control tool window.

Moreover, PhpStorm helps you analyze how the changes affect a model across revisions, and provides the action Compare all classes from revision on UML .

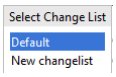
Changes in Class diagram are color-coded:

- Green for added elements.
- Gray for deleted elements.
- Blue for modified elements.



To view changes in UML Class diagram

1. In the Local Changes tab of the Version Control tool window, select the desired changelist from the suggested popup:

   Note that if there is only one changelist, the popup won;t show up.

2. Do one of the following:
   - In the toolbar of the Local tab, click , or press `Ctrl+Shift+Alt+D`.
   - On the context menu of the editor or the Project tool window, choose Diagrams | Show Changes, or press `Ctrl+Shift+Alt+D`.

   The diagram opens in a pop-up window. Double-click a node to view changes in a Differences viewer.

To view changes in revisions as UML Class diagram

1. In the History tab of the Version Control tool window , select the desired revision.

2. Click , or press `Ctrl+Shift+D`. The diagram opens in a pop-up window.

Viewing Class Hierarchy as a Class Diagram

PhpStorm helps you view the complete hierarchy of the selected type.

So doing, you can modify the contents of the UML Class diagram, to show ancestor or descendant classes, types used in method signatures, and perform all actions that are available for UML Class diagrams.

To view class hierarchy as a UML Class diagram

1. Open the desired class in the editor, and place the caret at the type you want to see hierarchy for.

2. Do one of the following:
   - On the context menu, point to Diagrams, and then choose Show Diagram, or Show Diagram Popup
   - Press `Ctrl+Shift+Alt+U` or `Ctrl+Alt+U`

Tip You can view type hierarchy for any class selected in the Project tool window:

Viewing Siblings and Children

PhpStorm can smartly distinguish nodes that belong to the same namespace from extraneous ones.

To highlight sibling nodes

1. Select a node element in diagram.

2. Start navigating through the diagram. So doing, extraneous classes are automatically grayed out.

Viewing Ancestors, Descendants, and Usages

- To show ancestor and descendant types
- To find usages of a diagram element

To show ancestor and descendant types

1. Open UML Class diagram
2. Select a node element in diagram.
3. On the context menu, choose one of the following commands, or press keyboard shortcuts:
   - Go to | Implementation `Ctrl+Alt+B`
   - Show Parents `Ctrl+Alt+P`
4. If there are several classes available, select the desired one from the pop-up list:
   The selected class will be added to the UML Class diagram, with the corresponding relationship links:

To find usages of a diagram element

1. Select a diagram element.
2. On the context menu of the selection, select the desired Find Usages command.

Using Blade Templates

On this page:

- Overview
- Preparing to use Blade templates
- Adding, editing, and removing Blade directives
- Configuring Blade delimiters

Overview

PhpStorm provides full support of the Laravel Blade template engine up to version 5.1. This support involves:

- Highlighting of Blade syntax in template files.
- Code completion for all Blade directives, both predefined and custom, as well as for braces.
- In `@for` and `@foreach` directives, variable introduction is offered with autocompletion inside the code constructs.
- Expanding and folding sections defined through the `@section` directive. A block of code between a `@section` directive and a closing directive (for example, `@stop`) can be expanded or folded by clicking or in the gutter area.
- Dedicated Blade-specific code inspections, for example an inspection to check that a section opened with `@section` directive is closed with one of the corresponding directives.
- Blade-aware navigation with `Ctrl+B` includes links to templates in `@extends` and `@include` directives.
- Finding usages (`Alt+F7`) can be invoked on a file name or a symbol in the code to show all the usages of a template across your codebase. Currently this functionality is available only inside other templates, but not from views.
- Customizing predefined Blade directives and defining custom directives for Blade templates.

Preparing to use Blade templates

Before you start, make sure the PHP and Blade plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Adding, editing, and removing Blade directives

In PhpStorm, Blade directives are managed in the Directives tab of the Blade Page. The tab lists all the currently available Blade directives, for those that have parameters, the prefixes and suffixes are also shown. When you start, the list contains only predefined directives. You can edit these directives as well as create custom ones.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Blade under PHP. On the Blade page that opens, switch to the Directives tab. The tab shows a list of all currently available directives.
2. Do one of the following:
   - To define a new directive, click the Add directive button and then specify the directive's name in the Name text box. If the new directives requires a prefix and a suffix, select the Has parameter check box and type the prefix and suffix to use in the Prefix and Suffix text boxes respectively. PhpStorm will automatically enclose the prefix and suffix in opening and closing brackets and quotes and add a colon separator `:` so the parameters will look as follows: `("<prefix>:<suffix>")` .
   - To edit an existing directive, select it in the list and change the values in the text boxes below. To restore the original definition, click the Reset to defaults button .
   - To remove a directive from the list, select it and click the Remove directive button .

Configuring Blade delimiters

PhpStorm recognizes Blade templates and provides error highlighting and code completion for them based on the delimiters you specify. These delimiters are managed in the Text Tags tab of the Blade Page .

The fields in the tab show the opening and closing characters for raw tags , content tags , and escaped tags .

The fields are filled in with the default values in compliance with Blade Templates 5.1 . If you are using an earlier version, you can specify the relevant custom delimiters and PhpStorm will provide coding assistance according to the new rules.

Using WordPress Content Management System

WordPress WordPress WordPress WordPress

You can run WordPress from PhpStorm in two modes:

- Through a dedicated user interface. To get access to this functionality in a project, you need to download WordPress, register it in PhpStorm, and activated the WordPress integration within the current project.
- Through a set of command line tools. In this case, you need to download WP-CLI and configure it as a PhpStorm command line tools, see Creating a Command Line Tool .

In this section:

- Preparing to Use WordPress
- WordPress Specific Coding Assistance
- Using the WordPress Command Line Tool WP-CLI

Preparing to Use WordPress

On this page:

- Overview
- Before you start
- Downloading and installing WordPress
- Activating the WordPress installation in a project
- Generating a WordPress application stub

Overview

PhpStorm provides a dedicated interface for developing and running WordPress applications and provides WordPress-aware coding assistance, see WordPress Specific Coding Assistance . To get access to this functionality in a project, you need to download WordPress, register it in PhpStorm, and activated the WordPress integration within the current project.

Alternatively, you can download WordPress and create a stub of a WordPress-targeted project. In this case, the WordPress integration in the project will be activated automatically.

Whether you enable the WordPress integration in an existing project or create a stub of a WordPress application, PhpStorm checks if the development environment is configured properly for WordPress development. If the configuration does not meet the requirements, PhpStorm displays a pop-up window with a Fix link.

Before you start

1. Make sure the PHP interpreter is configured in PhpStorm on the PHP page , as described in Enabling PHP Support .
2. Make sure the PHP and WordPress Support plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Configure the PHP interpreter to use WordPress with, as described Configuring Remote PHP Interpreters . Note that WordPress requires PHP 5.3 or higher.

Downloading and installing WordPress

1. Download the `WordPress.zip` archive at https://wordpress.org/download/ .
2. Extract the archive:
   - To have the possibility to run your WordPress application on a local Web server, store the extracted file to the document root of the server where the PHP interpreter is configured. For local development environment with the Apache HTTPD Web server, extract WordPress.zip to the `htdocs` folder.
   - To run your WordPress application on an in-place server, store the extracted file under the project root.
   - To run your WordPress application on a remote server, store the extracted files in your project and then configure automatic upload of them to the document root of the remote server.
     Learn more about server access configurations, see Configuring Synchronization with a Web Server . For information on configuring upload to the server, see Uploading and Downloading Files .
   - If you are not going to run your WordPress application but just need to get coding assistance from PhpStorm, store the extracted files anywhere on your computer. In this case, you will have to configure the installation as an include path, see Configuring Include Paths .

To run WordPress in the command line mode, you will need a set of command line tools which you can acquire by installing the `wp-cli/wp-cli` package using the Composer dependency manager or by downloading the `wp-cli.phar` archive. For details, see Using the WordPress Command Line Tool WP-CLI .

Activating the WordPress installation in a project

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click WordPress under Tools.
2. On the WordPress page that opens, select the Enable WordPress Integration check box.
3. In the WordPress Installation Path text box, specify the folder where WordPress is installed. This folder should contain the `wp-admin` and `wp-includes` subdirectories.
4. Click Apply to save the WordPress registration and click PHP.
5. On the PHP page that opens, add the path to the WordPress installation folder to the Include Paths list: click the + button and specify the path to the installation folder in the dialog box that opens. Learn more in Configuring Include Paths .

Generating a WordPress application stub

During project creation, PhpStorm can generate a project stub for developing WordPress applications. The project structure is set up in accordance with the WordPress requirements.

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose WordPress Plugin.
3. In the right-hand pane, specify the folder where the project will be created. In the WordPress Installation Path text box, specify the folder where WordPress is installed. This folder should contain the `wp-admin` and `wp-includes` subdirectories. Click Create to start the project stub generation.
4. If the newly created project is outside the WordPress installation folder, configure it as an external library by adding it to the list of included path.
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
   2. On the PHP page that opens, add the path to the WordPress installation folder to the Include Paths list: click the + button and specify the path to the installation folder in the dialog box that opens. Learn more in Configuring Include Paths .

WordPress Specific Coding Assistance

PhpStorm provides full coding assistance for developing WordPress including WordPress-aware code completion, search for hook registration functions and functions specified as hook registration parameters, navigation between hook registrations and the hook invocations, callbacks, the possibility to configure the coding style in accordance with the WordPress code style, viewing the official WordPress documentation from the PhpStorm editor, etc.

On this page:

- Adding the WordPress to a project
  - Configuring the WordPress installation as a project include path
  - Adding the wp-content folder to the project
- Configuring WordPress code style
- Hooks support
  - Completion for parameters of the Action and Filter functions
  - Navigation from a hook registration to the hook invocation
  - Callbacks from a hook registration
  - Navigating to hook invocations
  - Searching for hook registration functions
- Viewing the official WordPress documentation from PhpStorm

Adding the WordPress to a project

To take advantage of coding assistance provided by PhpStorm, the WordPress installation you are working with should be configured in the project as an external library. As a result, the WordPress core file will be involved in indexing which is the basis for resolving references and providing code completion, navigation, search. etc.

If the `wp-content` folder is outside the WordPress installation, it has to be added either as an include path or as a content root.

Configuring the WordPress installation as a project include path

To add WordPress to your project as an external library, you need to add it to the list of include paths. Learn more in Configuring Include Paths .

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.

2. On the PHP page that opens, add the path to the WordPress installation folder to the Include Paths list: click the + button and specify the path to the installation folder in the dialog box that opens.

Adding the wp-content folder to the project

If the `wp-content` folder is located outside the WordPress installation, you need to add it to the project individually, apart from the WordPress core files. This can be done in two ways:

- To have the `wp-content` folder involved in indexing without putting it under the project version control, add it to the list of include paths:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click PHP under Languages&Frameworks.
  2. On the PHP page that opens, add the path to the `wp-content` folder to the Include Paths list: click the + button and specify the path to the folder in the dialog box that opens.

- To have the `wp-content` folder involved in indexing and put it under the project version control, add it as a content root:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Directories under Project:<project name>.
  2. On the Directories page that opens, click the Add Content Root button +. In the dialog that opens , locate the `wp-content` directory and click OK.

  Learn more about configuring content roots in Configuring Content Roots . For information about adding files and folders to version control, see Enabling Version Control .

Configuring WordPress code style

In PhpStorm, you can use the WordPress Code Style in accordance with the WordPress coding standards . To configure the native WordPress code style, do one of the following:

- 1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click PHP under Code Style.
  2. On the Code Style. PHP page that opens, click the Set from... link.
  3. On the context menu, choose Predefined Style, then choose WordPress.

- Upon activating of the WordPress integration, PhpStorm displays a pop-up window offering you to set the WordPress code style. Click the Set it link in the pop-up window and set the code style on the Code Style. PHP page that opens.

Learn more in Configuring Code Style .

Hooks support

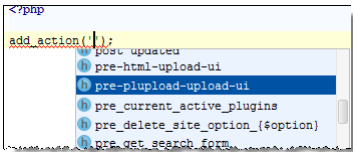
PhpStorm indexes all hooks declared in the WordPress core and in the included plugins. This makes the basis for coding assistance when working with hooks.

Completion for parameters of the Action and Filter functions

Hook names are available in code completion of standard parameters for action and filter functions (`add_action()` and `add_filter()`). To invoke completion for a parameter:

1. Start typing the declaration of an `add_action()` or `add_filter()` function.

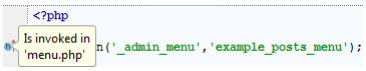
2. Press `Ctrl+Space` and choose the relevant parameter from the list.



Navigation from a hook registration to the hook invocation

To navigate from a hook registration (`add_action()` or `add_filter()` function) to the hook invocation:

– Click the icon in the gutter area next to the hook registration to navigate from.



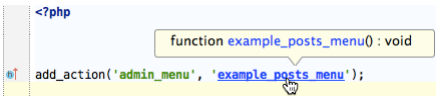
PhpStorm opens the file where the hook is invoked and positions the cursor at the invocation method, in the current example it is `do_action` .

>

Callbacks from a hook registration

You can navigate to the declaration of the function or method specified as the second parameter of a hook registration (`add_action()` or `add_filter()` function). To do that:

– With the `Ctrl` keyboard key pressed, hover the cursor over the parameter of interest. PhpStorm displays a pop-up information message with the definition of the function or method specified as this parameter:



The parameter turns into a link. When you click this link, PhpStorm opens the file where the function or method was declared and positions the cursor and declaration.

>

Navigating to hook invocations

To search for a hook invocation and navigate to it, do one of the following:

– Use the Navigate to Symbol functionality:

1. Choose Navigate | Symbol on the main menu.
2. In the Enter symbol name pop-up window that appears, start typing the hook name in the search field. If necessary, select the Include non-project items check box. The contents of the list below the search field change as you type.
3. Click the relevant hook from the list. %product" opens the file where it is invoked and positions the cursor at the invocation.

– Use the Search Everywhere functionality:

1. Press the Shift keyboard key twice.
2. start typing the hook name in the search field. If necessary, select the Include non-project items check box. The contents of the list below the search field change as you type.
3. Click the relevant hook from the list. %product" opens the file where it is invoked and positions the cursor at the invocation.

Learn more about search and navigation in Searching Through the Source Code and Navigating Through the Source Code .

Searching for hook registration functions

You can search for occurrences of a hook registration function and for occurrences of a function or method specified as the second parameter of a hook registration.

– To get a list of usages for a hook registration function:

1. Position the cursor at the name of the desired hook registration function and press `Alt+F7` or choose Edit | Find | Find Usages on the main menu.
2. From the pop-up list that shows the available options in terms of hook/function name, choose the relevant option. The Find tool window opens with a new tab showing all the detected occurrences of the selected function.
3. Explore the listed occurrences. To navigate to the relevant one, click it. PhpStorm opens the file with the selected function or method and positions the cursor at it.

– To get a list of occurrences of a function or method specified as the second parameter of a hook registration:

1. Position the cursor at the parameter of interest and press `Alt+F7` or choose Edit | Find | Find Usages on the main menu. The Find tool window opens with a new tab showing all the detected occurrences of the function or method specified as the parameter on which the search was invoked.
2. Explore the listed occurrences. To navigate to the relevant one, click it. PhpStorm opens the file with the selected function or method and positions the cursor at it.

Learn more in Finding Usages .

Viewing the official WordPress documentation from PhpStorm

You can view the official WordPress Documentation at http://wordpress.org/ right from the PhpStorm editor.

Select the text you are interested in

1. Select the text you are interested in.
2. On the context menu of the selection, choose Search on WordPress.org.
   PhpStorm opens the page with the relevant documentation in the default browser.

Using the WordPress Command Line Tool WP-CLI

To run WordPress in the command line mode, you will need a set of command line tools which you can acquire by installing the `wp-cli/wp-cli` package using the Composer dependency manager or by downloading the `wp-cli.phar` archive. The downloaded command line tool must be registered in PhpStorm as described in Using Command Line Tools .

For information about running the tool in the command line mode, see Running Command Line Tool Commands .

On this page:

– Before you start
– Installing the wp-cli package using the Composer dependency manager
– Downloading the wp-cli.phar archive
– Configuring wp-cli as a PhpStorm command line tool
– Running the wp-cli tool

Before you start

1. Make sure the PHP interpreter is configured in PhpStorm on the PHP page , as described in Enabling PHP Support .
2. Make sure the PHP and Command Line Tool Support plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Configure the PHP interpreter to use WordPress with, as described Configuring Remote PHP Interpreters . Note that WordPress requires PHP 5.3 or higher.

Installing the wp-cli package using the Composer dependency manager

Before you start, make sure Composer is installed on your machine and initialized in the current project, see Using Composer Dependency Manager .

1. On the main menu, choose Tools | Composer | Add Dependency.
2. In the Add Composer Dependency dialog that opens, select the `wp-cli/wp-cli` package from the Available Packages list, possibly using the search field. The list shows all the available packages, however, the packages that have been already installed are marked with a tick.
   Choose the relevant version from the Version to install drop-down list.
3. If necessary, expand the Settings hidden area and specify the advanced installation options. In the Command line parameters text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have the package added to the `require-dev` section instead of the default `require` section type `-dev` . For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .
4. Click Install.

When you click Create, the `create-project` command is invoked with the selected package. This results in creating a Composer project whose configuration and structure depends on the selected package, see https://getcomposer.org/doc/03-cli.md#create-project for details. After that a PhpStorm project opens.

Downloading the wp-cli.phar archive

– Download the `wp-cli.phar` at http://wp-cli.org/ .

Configuring wp-cli as a PhpStorm command line tool

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Command Line Tool Support under Tools.
2. On the Command Line Tool Support page, click the Add button. In the Choose Tool to Add dialog box that opens, choose WP-CLI.
3. In the WP-CLI dialog box that opens, choose the way to run WordPress:
   – Installed via PHAR: Choose this option to launch WordPress through a PHP script or have PhpStorm detect and start the launcher in the `wp-cli.phar` archive.
   In the Path to PHP executable field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.

   In the Path to phar text box, specify the location of the `wp-cli.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
   – Executable available (installed via Composer, etc.): choose this option to launch WordPress through an executable file which is available when you install WordPress using a package management tool, for example, Composer.
   In the Path to wp.bat field, specify the location of the `wp.bat` or `wp` executable file. If you used Composer, the default location is `\vendor\wp\cli\bin\wp` or `\vendor\wp\cli\bin\wp.bat` . Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
4. When you click OK, the WP-CLI dialog box closes and PhpStorm brings you back to the Command Line Tool Support page, where wp is added to the list of available tools.
5. From the Console encoding drop-down list, choose the character set to show the tool's output in the Command Line Tools Console Tool Window .
6. In the Show console in area, specify where you want the Input pane for typing commands opened:
   – To have the Input pane opened in a pop-up window, choose the Pop-up option.
   – To have the Input pane opened as a text box at the bottom of the Command Line Tools Console tool window, choose the Tool window option.

Running the wp-cli tool

You can run WP-CLI commands and analyze their output right in PhpStorm using the dedicated Command Line Tools Console tool window, just as the commands of any

other command line tools, see Running Command Line Tool Commands .

Using Google App Engine for PHP

With PhpStorm, you can develop a PHP application for running in the Google PHP Runtime Environment . PhpStorm supports all the major Google App Engine for PHP development practices and provides the possibility to run and debug your applications locally before uploading them to the runtime environment.

In this section:

- Preparing to Develop a Google App for PHP Application
- Running, Debugging, and Uploading an Application to Google App Engine for PHP

Google App Engine support in PhpStorm

- A dedicated project type with a specific directory structure and configuration file.
- Generating and analyzing a model dependency diagram for the Google App Engine models.
- A dedicated App Engine for PHP run/debug configuration for running and debugging applications on the PHP Development Server which comes bundled with the Google App Engine for PHP SDK.
- Uploading applications using the command of the Tools menu.

Preparing to Develop a Google App for PHP Application

PhpStorm takes care of creating the specific directory structure and settings. PhpStorm can either generate a Google App Engine for PHP-specific project stub or you can enable support of Google App Engine for PHP in an existing project.

- Before you start
- Creating a project stub of a Google App Engine for PHP application
- Activating the Google App Engine support in an existing project

Before you start

1. Download and install Python , version 2.7.
2. Download and install the Google App Engine for PHP .
3. Make sure the Google App Engine Support for PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
4. Create a Google account at http://www.google.com .
5. Create an application on the Create application page of the Google App Engine and remember its application ID, you will have to specify it during the creation of a project.

Creating a project stub of a Google App Engine for PHP application

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the Create New Project dialog box that opens, specify the name of the project and the folder where the project will be located. From the Project Type drop-down list, choose App Engine Project and click OK.
3. In the right-hand pane, specify the following:
   - In the Application ID text box, type the identifier of your application as you specified it on the Create application page.
   - In the SDK directory test box,
   - The location of the Python executable file. Type the path manually in the Python executable text box or click the Browse button ... and choose the Python executable in the dialog box that opens.

PhpStorm creates a project with the `main.php` file and the app.yaml configuration file. The `app.yaml` file contains the generated settings for the runtime (should be `php55`), the api_version , the applicationID , the application version (by default is 1), the threadsafe element with the value `true` to enable sending multiple, parallel requests, and the Script handlers `url` and `script` (with the value `main.php`).

Activating the Google App Engine support in an existing project

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Google App Engine for PHP under PHP.
2. On the Google App Engine for PHP page that opens, select the Enable Google App Engine for PHP support check box and specify the following:
   1. The folder where the Google App Engine for PHP SDK is stored.
   2. The path to the Python 2.7 executable file.
   3. In the App Engine Account Settings area, choose the way to authenticate to the development server, the available options are:
      - Use passwordless login via OAuth2: choose this option to use the OAuth 2.0 protocol . To save the token achieved through the Google Developers Console , clear the Do not save token check box.
      - Log in with email and password: choose this option to use your Gmail address and password.

Running, Debugging, and Uploading an Application to Google App Engine for PHP

PhpStorm provides the possibility to run and debug your applications locally on the PHP Development Server before uploading them to the Google PHP Runtime Environment . Applications are launched locally according to a dedicated App Engine for PHP run/debug configuration.

On this page:

- Working with the app.yaml file
- Creating a run configuration fo the type App Engine for PHP
- Running an application on the PHP development server
- Debugging an application on the PHP development server
- Uploading an application to the Google PHP runtime environment

Working with the app.yaml file

A Google App Engine PHP application is configured in the app.yaml file. If you generate a project stub (see Preparing to Develop a Google App for PHP Application), the `app.yaml` file is created automatically. If you activate Google App Engine support in an existing project, you need to create an `app.yaml` file manually. In either case, make sure the `runtime` property is set to `php55` , because the Google App Engine for PHP SDK contains a built-in PHP 5.5 interpreter.

Creating a run configuration fo the type App Engine for PHP

1. On the main menu, choose Run | Edit Configurations.
2. Click the Add New Configuration toolbar button +, and choose App Engine for PHP on the context menu.
3. In the Run/Debug Configuration: App Engine for PHP dialog box that opens, specify the host to run the development server and the application on, the default is `localhost` , and the port through which PhpStorm will communicate with the development server, the default port is `8080` .
4. Optionally in the Command Line area, specify the settings for running and debugging your application on the PHP development server in the command-line mode :
   - In the Interpreter options field, specify the options to be passed to the PHP executable file of the built-in PHP interpreter, see Command-Line Arguments for details.
   - In the Custom working directory field, specify the location of the files that are outside the folder with your sources and are referenced through relative paths. Type the path manually or click the Browse button ... and select the desired folder in the dialog that opens .
   - In the Yaml files field, specify the `.yaml` configuration files to use. This field is optional, use it when your application consists of several modules and therefore several `.yaml` configuration files are used.
   - In the Environment variables field, In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Running an application on the PHP development server

1. Create an App Engine for PHP run configuration as described above.
2. Choose the configuration from the list and click ▶ on the toolbar.
3. View and analyze the output of the application in console of the Run tool window .
4. To view the application execution results, open your browser at `http://localhost:8080` .

Debugging an application on the PHP development server

1. Create an App Engine for PHP run configuration as described above.
2. Set the breakpoints in your code, see Using Breakpoints .
3. Choose the configuration from the list and click 🐞
4. In the Debug tool window that opens, step through the program , stop and resume the program, examine it when suspended, etc. See Stepping Through the Program , Pausing and Resuming the Debugger Session , and Examining Suspended Program for details. ,

Uploading an application to the Google PHP runtime environment

After you have tested your application by running and debugging it locally on the development server, you can deploy it to the Google PHP Runtime Environment .

1. On the main menu, choose Tools | Google App Engine for PHP | Upload App Engine PHP app.
2. Visit the application at `http://<your-application-id>.appspot.com/` .

CoffeeScript Support

In this section:

- CoffeeScript Support
  - Overview
  - Preparing for CoffeeScript development
  - Coding assistance
- Transpiling CoffeeScript to JavaScript
- Running CoffeeScript
- Debugging CoffeeScript

Overview

PhpStorm provides CoffeeScript support. PhpStorm recognizes `*.coffee` files, and allows you to edit them providing full range of coding assistance without any additional steps from your side. CoffeeScript files are marked with icon.

To run, debug, and test your code, you will need it translated into JavaScript which requires a transpiler and Node.js. For more details on CoffeeScript transpilation, see Transpiling CoffeeScript to JavaScript .

Preparing for CoffeeScript development

1. Make sure the CoffeeScript and Node.js plugins are installed and enabled.

   These plugins are not bundled with PhpStorm, but they are available from the PhpStorm plugin repository . Once enabled, the plugins are available at the IDE level, that is, you can use them in all your PhpStorm projects. See Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins for details.
2. Download and install the Node.js framework.
3. Configure the Node.js interpreter in PhpStorm:
   2. On the Node.js and NPM page that opens, specify the location of the desired Node.js interpreter.

Coding assistance

CoffeeScript support includes:

- Code completion for keywords, labels, variables, parameters and functions.
- Error and syntax highlighting.
- Code formatting and folding .
- Refactoring:
  - Common refactoring procedures, such as extract method , inline, rename/move, etc.
  - CoffeeScript-specific refactoring procedures, such as change signature, extract parameter, extract variable.

  See JavaScript-Specific Refactorings for details.
- Code generation
  - Generating code stubs based on file templates during file creation.
  - Ability to create line and block comments (`Ctrl+Slash` / `Ctrl+Shift+Slash`).
- Navigation through source code
  - Navigating with Structure View .
  - Navigate | Declaration (`Ctrl+B`).
  - Navigate | Implementation (`Ctrl+Alt+B`) from overridden method / subclassed class.
  - Navigate | Symbol (`Ctrl+Shift+Alt+N`).
- Compiling to JavaScript for further running, debugging, and testing, see Running CoffeeScript and Debugging CoffeeScript .
- Executing CoffeeScript files involves:
  - Ability to preview results of CoffeeScript files compilation to JavaScript.
  - Ability to launch CoffeeScript files from the context menu.
  - Run/debug configuration for NodeJS includes the ability to use CoffeeScript plugin.

Transpiling CoffeeScript to JavaScript

On this page:

- Introduction
- Before you start
- Installing the CoffeeScript transpiler globally
- Installing the CoffeeScript transpiler in a project
- Creating a File Watcher
- Examples of customizing the behaviour of a transpiler
- Transpiling the CoffeeScript code
- Previewing the transpilation results without running a transpiler

Introduction

CoffeeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, CoffeeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

PhpStorm supports integration with the coffee-script transpilation tool. The tool translates CoffeeScript code into JavaScript and creates source maps that set correspondence between lines in your CoffeeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.

In PhpStorm, transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

The easiest way to install the CoffeeScript transpiler is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the CoffeeScript transpiler executable file, choose one of the following methods:

- Install the transpiler globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the transpiler in a specific project and thus restrict its use to this project.
- Install the transpiler in a project as a development dependency .

In either installation mode, make sure that the parent folder of the CoffeeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

Before you start

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The CoffeeScript transpiler is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the CoffeeScript transpiler.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the CoffeeScript transpiler and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.

Installing the CoffeeScript transpiler globally

Global installation makes a transpiler available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the transpiler is automatically added to the `PATH` variable, which enables you to launch the transpiler from any folder. To install the transpiler globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g coffee-script
     ```

     The `-g` key makes the transpiler run in the global mode. Because the installation is performed through NPM, the CoffeeScript transpiler is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the CoffeeScript transpiler, see https://npmjs.org/package/coffee-script .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing the CoffeeScript transpiler in a project

Installing a transpiler in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install coffee-script
```

- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package.
  4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the CoffeeScript transpiler in a `node_modules` folder. If your project already contains such folder, the CoffeeScript transpiler is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the CoffeeScript transpiler in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the CoffeeScript transpiler is installed there.

In either case, make sure that the parent folder of the CoffeeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

Creating a File Watcher

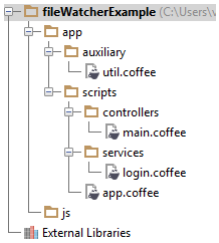
PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
2. Click the Add button + or press `Alt+Insert` and choose the CoffeeScript predefined template from the pop-up list. Your code will be translated to JavaScript and supplied with generated source maps .
3. In the Program text box, specify the path to the `coffee.cmd` file. Type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.
4. Proceed as described on page Using File Watchers .

Examples of customizing the behaviour of a transpiler

Any transpiler is an external, third-party tool. Therefore the only way to influence a transpiler is pass arguments to it just as if you were working in the command line mode. Below are two examples of customizing the default output location for the CoffeeScript transpiler.

Suppose, you have a project with the following folder structure:



By default, the generated files will be stored in the folder where the original file is. You can change this default location and have the generated files stored in the `js` folder. Moreover, you can have them stored in a flat list or arranged in the folder structure that repeats the original structure under the `app` node.

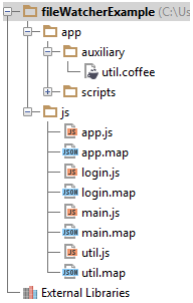
- To have all the generated files stored in the output `js` folder without retaining the original folder structure under the `app` folder:
  1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\ --compile --map $FileName$
```

  2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileNameWithoutExtension$.map
```

As a result, the project tree looks as follows:



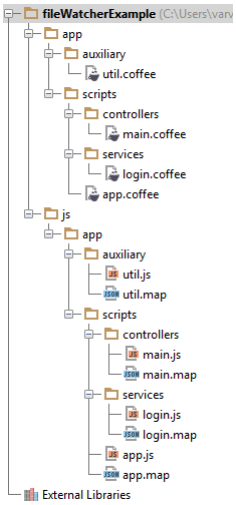
- To have the original folder structure under the `app` node retained in the output `js` folder:
  1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\ --compile --map $FileName$
```

  2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileDirRelativeToI
```

As a result, the project tree looks as follows:



Transpiling the CoffeeScript code

When you open a CoffeeScript file, PhpStorm checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, PhpStorm displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, PhpStorm starts it automatically upon the event specified in the New Watcher dialog .

- If the Immediate file synchronization check box is selected, the File Watcher is invoked as soon as any changes are made to the source code.
- If the Immediate file synchronization check box is cleared, the File Watcher is started upon save (File | Save All, `Ctrl+S`) or when you move focus from PhpStorm (upon frame deactivation).

The transpiler stores the generated output in a separate file. The file has the name of the source CoffeeScript file and the extension `js` or `js.map` depending on the transpiler type. The location of the generated files is defined in the Output paths to refresh text box of the New Watcher dialog . Based on this setting, PhpStorm detects the transpiler output. However, in the Project Tree, they are shown under the source `.coffee` file which is now displayed as a node.

Previewing the transpilation results without running a transpiler

PhpStorm can perform static analyses of your CoffeeScript code without actually running a transpiler and display the predicted transpilation output in the dedicated read-only viewer.

1. Open the desired CoffeeScript file in the editor, and right-click the editor background.
2. On the context menu, choose Preview Compiled CoffeeScript File. The preview is opened in the dedicated read-only viewer: the left-hand pane shows the original CoffeeScript source code and the right-hand pane shows the JavaScript code that will be generated by the transpiler when it runs.

Running CoffeeScript

On this page:

- Introduction
- Transpiling CoffeeScript manually and running the generated JavaScript code
- Transpile CoffeeScript on the fly during run

Introduction

CoffeeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, CoffeeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

For more details about transpilation in PhpStorm, see the section Using File Watchers .

In either case, running CoffeeScript is supported only in the local mode. This means that PhpStorm itself starts the Node.js engine and the target application according to a run configuration and gets full control over the session.

For more details about running Node.js applications, see Running and Debugging Node.js .

There are two approaches to running CoffeeScript in PhpStorm:

- Transpile the CoffeeScript code manually and then run the output JavaScript code as if it were a Node.js application.
- Run the original CoffeeScript code through the NodeJS run configuration and have PhpStorm transpile it on the fly.

Transpiling CoffeeScript manually and running the generated JavaScript code

1. Transpile the CoffeeScript code into Javascript .
2. Start creating a Node.js run configuration with the following mandatory settings:
   1. The Node.js engine to use. By default, the field shows the path to the interpreter specified on the Node.js page during Node.js configuration.
   2. In the Working directory field, specify the location of the files referenced from the starting CoffeeScript file to run, for example, includes. If this file does not reference any other files, just leave the field empty.
   3. In the Path to Node App JS File text box, specify the full path to the JavaScript file that was generated from the original CoffeeScript file during the transpilation.
3. Save the configuration and click ▶ on the toolbar.
4. Proceed as while running a Node.js application .

Transpile CoffeeScript on the fly during run

1. This mode requires that the `register.js` file, which is a part of the `coffee-script` package, should be located inside the project. Therefore you need to install the `coffee-script` package on the Node.js page locally, as described in Installing and Removing External Software Using Node Package Manager .
2. Open the starting CoffeeScript file in the editor or select in the Project tool window and choose Create <CoffecScript_file_name> on the context menu. Alternatively, start creating a Node.js run configuration as described in Running and Debugging Node.js . In the Run/Debug Configuration: Node JS dialog that opens, specify the following mandatory settings:
   1. The Node interpreter to use. Select the relevant interpreter configuration or create a new one, see By default, the field shows the path to the interpreter specified on the Node.js page during Node.js configuration.
      For Linux and OS X, this setting is overridden by the Node.js from the path to the CoffeeScript transpiler executable file.
   2. In the Node parameters text box, type the following:

      ```
      --require coffee-script/register
      ```

   3. In the Working directory field, specify the working directory of the application. All references in the starting CoffeeScript file, for example, imports, will be resolved relative to this folder, unless such references use full paths.
      By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
   4. In the JavaScript file text box, specify the full path to the CoffeeScript file to run.

   Note that all the mandatory fields will be filled in automatically if you create a run configuration directly from the required CoffeeScript file.
3. Save the configuration and click ▶ on the toolbar.
4. Proceed as while running a Node.js application .

Debugging CoffeeScript

CoffeeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, CoffeeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

To debug CoffeeScript in PhpStorm, you need source maps generated in addition to the JavaScript code. Source maps set correspondence between lines in your CoffeeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly. JavaScript and source maps are generated by transpiling the CoffeeScript code manually using the File Watcher of the type CoffeeScript. After that you can debug the output JavaScript code as if it were a Node.js application.

For more details about transpilation in PhpStorm, see the section Using File Watchers .

Debugging CoffeeScript is supported only in the local mode. This means that PhpStorm itself starts the Node.js engine and the target application according to a run configuration and gets full control over the session.

For more details about debugging Node.js applications, see Running and Debugging Node.js .

1. Set the breakpoints in the CoffeeScript code, where necessary.
2. Transpile the CoffeeScript code into Javascript using the File Watcher of the type CoffeeScript Source Map.
3. Start creating a Node.js run configuration with the following mandatory settings:
   1. The Node.js engine to use. By default, the field shows the path to the interpreter specified on the Node.js page during Node.js configuration.
   2. In the Working directory field, specify the location of the files referenced from the starting CoffeeScript file to run, for example, includes. If this file does not reference any other files, just leave the field empty.
   3. In the Path to Node App JS File text box, specify the full path to the JavaScript file that was generated from the original CoffeeScript file during the transpilation.
4. Save the configuration and click on the toolbar.
5. Proceed as while debugging a Node.js application locally .

Testing CoffeeScript

To run unit CoffeeScript unit test, you need to use dedicated packages that

Copyright

- Copyright
  - Prerequisite
  - Basics
- Associating a Copyright Profile with a Scope
- Generating and Updating Copyright Notice

Prerequisite

Before using copyright profiles and generating copyright notices, make sure that the Copyright plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Basics

Your project can contain files to be protected by copyright. Instead of generating and editing copyright notices in each file separately, PhpStorm suggests that you use copyright profiles. Each copyright profile is associated with a certain scope and defines the text and formatting of the copyright notice to be inserted in the files within this scope.

You may need several different copyright notices for files within the same project. In this case, PhpStorm enables you to define several profiles and associate them with relevant scopes.

You can also declare one of the available profiles as default project copyright profile. The settings of this profile will be applied to any project file that is not included in any scope or belongs to a scope, which has no associated copyright profile .

A copyright profile can contain an explicit plain text of the copyright notice or its definition through a Velocity template. You can type the desired text or a Velocity template manually or import a copyright notice definition from an existing profile.

Note the following:

- If a file does not belong to any scope with a copyright profile assigned, the default copyright profile is used.
- If the No copyright option has been set as the Default Copyright Profile in the Copyright settings dialog box, no copyright notice will be created or updated and PhpStorm will prompt you to configure a profile to apply.

Associating a Copyright Profile with a Scope

On this page:

– Associating a profile with a scope

– Setting the default Copyright profile for a project

Associating a profile with a scope

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Copyright under Editor. The Copyright page that opens displays mappings between the available copyright profiles and scopes.
   Note that unless you have at least one copyright profile and scope , the controls of the Scope-profile combinations area are disabled.
2. Click + (`Alt+Insert`). A row to define a new mapping is created.
3. From the Scope drop-down list, select the desired scope.
   To define a new scope, open the Scopes page or click the link in the lower part of the Copyright page.
4. From the Copyright drop-down list, select the profile to apply to the selected scope.
5. Continue composing the list of scope-profile mappings. Use + (`Alt+Insert`) or – (`Alt+Delete`) to add or remove the items. To reorder the mappings, use ↑ (`Alt+Up`) and ↓ (`Alt+Down`).

Setting the default Copyright profile for a project

The settings of the default profile will be applied to any project file that is not included in any scope or belongs to a scope that has no associated copyright profile.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Copyright under Editor.
2. On the Copyright page that opens, select the desired copyright profile from the Default project copyright drop-down list.
   Note that is you select No copyright, any attempt to generate a copyright notice will fail and PhpStorm will prompt you to create a profile to apply.

Generating and Updating Copyright Notice

In this section:

- Prerequisite
- Creating a Copyright profile from scratch
- Creating a Copyright profile based on the settings of an existing profile
- Importing an existing Copyright notice text
- Discarding a Copyright profile
- Adding a Copyright notice
- Updating a Copyright notice

Prerequisite

Before using copyright profiles and generating copyright notices, make sure that the Copyright plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Creating a Copyright profile from scratch

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Copyright Profiles under Copyright. The Copyright Profiles page opens.
2. Click the Add button + on the toolbar in the Create new copyright profile dialog box that opens specify the name of the new profile and click OK. You return to the Copyright Profiles page where the new profile is added to the list of available copyright profiles.
3. Specify the copyright notice to be generated. Do one of the following:
   - Type the desired plain text.
   - Define a Velocity template , then click the Validate button to check that the template has been specified correctly. Find the list of supported Velocity variables in the Copyright Profiles dialog box reference.
4. In the Regexp to detect copyright in comments text box, type a character string to distinguish copyright notices from other comments during copyright update.

   Tip Make sure that this regular expression matches the above specified copyright notice. Otherwise instead of updating copyright notices, PhpStorm will insert new ones.

Creating a Copyright profile based on the settings of an existing profile

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Copyright Profiles under Copyright. The Copyright Profiles page opens.
2. Select the desired profile to inherit the settings from and click the Copy button on the toolbar.
3. In the Create new copyright profile dialog box that opens specify the name of the new profile and click OK. You return to the Copyright Profiles page where the new profile is added to the list of available copyright profiles.
4. View and edit the profile settings. Proceed as during creation of a profile from scratch .

Importing an existing Copyright notice text

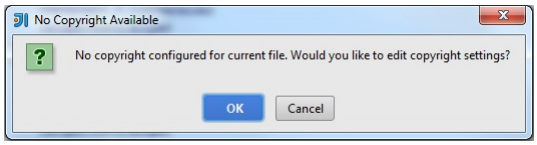
1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Copyright Profiles under Copyright. The Copyright Profiles page opens.
2. Click the Import button on the toolbar.
3. In the dialog that opens , choose the location of the `%.ipr%` file that refers to the copyright profile with the desired notice definition.
4. From the Choose profile to import list, that appears, select the desired copyright profile.
5. In the Create new copyright profile dialog box that opens specify the name of the new profile and click OK. You return to the Copyright Profiles page where the new profile is added to the list of available copyright profiles.
6. Proceed as during creation of a profile from scratch .

Discarding a Copyright profile

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click Copyright Profiles under Copyright. The Copyright Profiles page opens.
2. Select the profile you want to discard and click the Delete button — on the toolbar.

Adding a Copyright notice

1. Open the desired file in the editor.
2. Press `Alt+Insert`.
3. From the list that is displayed in the editor, choose Copyright.
4. The following steps depend on the belonging of a file to a certain scope and the value of the Default project copyright field in the Copyright page.
   - If the file where you want to generate a copyright notice belongs to a certain scope, the copyright notice is generated according to the copyright profile settings.
   - If the field Default project copyright has the value No copyright and the file where you want to generate a copyright notice does not belong to any scope, PhpStorm prompts you to configure copyright settings:

     Click OK to open the Copyright page and configure the desired settings.

Note that if a file belongs to several scopes, these scopes are checked upside down. So doing, the first scope with this file is declared the proper one, and its copyright profile is used.

Updating a Copyright notice

1. In the Project tool window, select files or directories where you want the copyright notice to be updated or just open the desired file in the editor.
2. Right-click the selection, and choose Update Copyright on the context menu.

   To have copyright notices updated correctly, make sure that the keyword specified in the copyright profile is a part of the notice definition. Otherwise, PhpStorm will not detect copyright notices and, instead of updating existing notices, will insert new ones.

> **Tip** You can also update copyright notices when committing changes .

Dart Support

In this section:

- Dart Support
  - Overview
- Preparing for Dart Development
- Using Integration with the Pub Tool
- Using Integration with the Dart Analysis Server
- Running and Debugging Dart Applications
- Testing Dart

Overview

PhpStorm supports developing, running, and debugging Dart code. PhpStorm recognizes `*.dart` files, and allows you to edit them providing full range of coding assistance. Dart files are marked with the icon.

Dart is intended for developing Web and command-line applications. PhpStorm supports both workflows.

- For running and debugging Dart command-line applications, PhpStorm provides a dedicated run/debug configuration of the type Dart Command Line Application.
- To run and debug a Dart Web application in a browser, you either need to use the Dartium browser or have the Dart code translated into JavaScript.
  Most browsers that work with JavaScript code do not natively process Dart code. The only exception is Dartium . This dart-oriented version of Chromium contains a built-in Dart VM and therefore can run Dart code directly. To be executed in any other browser, Dart code has to be compiled into JavaScript.

  With the pub tool, the compilation is performed through the pub serve action using the built-in dart2js compiler.

  You can run Dart Web application in any browser, while debugging is supported only in Dartium and Chrome. To run a Dart Web application you just need to open the starling HTML page in the browser of your choice. Debugging a Dart Web application is initiated through a run configuration of the type JavaScript Debug.

When the Dart development support is enabled, see Preparing for Dart Development , PhpStorm provides full coding assistance, which involves:

- Code completion for keywords, labels, variables, parameters and functions.
- Error and syntax highlighting.
- Code formatting and folding .
- Numerous code inspections and quick-fixes .
- Integration with the native Dart static analysis engine: all inconsistencies and potential problems are reported in the dedicated Dart Analysis Tool Window with the possibility to navigate to the fragment of the source code where the problem was detected.
- Refactoring:
  - Common refactoring procedures, such as extract method , inline, rename/move, etc.
  - Dart-specific refactoring procedures, such as change signature, extract parameter, extract variable.

  See JavaScript-Specific Refactorings for details.
- Code generation:
  - Generating code stubs based on file templates during file creation.
  - Ability to create line and block comments (`Ctrl+Slash` / `Ctrl+Shift+Slash`).
- Navigation and search through the source code:
  - Navigating with Structure View .
  - Navigate | Declaration (`Ctrl+B`).
  - Navigate | Implementation (`Ctrl+Alt+B`) from overridden method / subclassed class.

Preparing for Dart Development

On this page:

- Before you start
- Downloading the Dart tools
- Configuring Dart support in a project
  - Creating a Dart application stub
  - Configuring Dart support in an existing project
  - Developing several Dart projects (packages) within one PhpStorm project

Before you start

Install and enable the Dart plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Downloading the Dart tools

To have all the tools for developing both server-side and client-side applications, download the Dart SDK and the Dartium browser in which you can run Dart code without previously transpiling it into JavaScript. The Dart SDK incorporates the Dart Virtual Machine, the Dart Libraries, and all the background command line tools, the most important of them being the dart2js transpiler and the Dart-specific pub package manager .

Learn more about Dart development tools at the Dart Download Page .

Configuring Dart support in a project

To develop a Dart application you need to tell PhpStorm where the Dart SDK and the Dartium executable file are located. If you are using the Dart project template, PhpStorm will ask you to specify these settings during the project creation. Alternatively you can specify these paths at any time later on the Dart page of the Settings dialog box.

These settings are specific for each project. This means that you can have several Dart SDKs and Dartium installations in PhpStorm and switch between them from one project to another.

Next, you need a project with Dart-specific structure. If you are using the Dart project template, PhpStorm establishes the required folder structure for you automatically. Alternatively, you can add and arrange the necessary folders manually.

Creating a Dart application stub

During project creation, PhpStorm can generate a stub for developing Dart applications. The project structure is set up and some sources are generated based on the external Dart templates downloaded upon your request.

1. In the left-hand pane, choose Dart.
2. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the paths to the Dart SDK and to the Dartium executable file. PhpStorm detects and displays the Dart version. If the SDK was downloaded within the Dart Editor Bundle, the path to Dartium is also detected and filled in automatically.
   3. To have annotations verified and assertions checked, select the Start Dartium in checked mode check box. Learn more about Dartium command line flags at https://www.dartlang.org/tools/dartium/ .

   Click Create when ready.

If necessary, you can open the Dart page of the Settings dialog box by clicking the Settings button and configure command-line options and a custom user data directory for Dartium.

Configuring Dart support in an existing project

Besides creating a Dart project from the very beginning, you can configure Dart support in a project at any stage of its development.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Dart under Languages & Frameworks.
2. Select the Enable Dart support in project <project name> check box whereupon all the fields below become available.
3. In the Dart SDK Path text box, specify the location of the downloaded Dart SDK. Type the path manually or click the Browse button ... and choose the path in the dialog box that opens.
   If PhpStorm recognizes the Dart SDK correctly, it displays the revision number of the detected SDK in the Version read-only field.
4. In the Dartium Path text box, specify the location of the Dartium executable file. In the Dartium browser , you can run Dart code without previously transpiling it into JavaScript. Type the path manually or click the Browse button ... and choose the path in the dialog box that opens. If the SDK was downloaded within the Dart Editor Bundle, the path to Dartium is also detected and filled in automatically. Learn more about Dart development tools at the Dart Download Page .
5. To have annotations verified and assertions checked, select the Start Dartium in checked mode check box. Learn more about Dartium command line flags at https://www.dartlang.org/tools/dartium/ .

Developing several Dart projects (packages) within one PhpStorm project

If your application consists of several Dart projects (packages), you can develop all of them within one single PhpStorm project. To attach a Dart project (package) to an existing PhpStorm project you need to add its root folder as a content root.

To add a Dart project (package) to an existing PhpStorm project:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Directories under Project: <project name>.
2. In the right-hand pane which shows a list of already existing content roots, click Add Content Root. Then in the dialog box that opens, choose the root folder of the Dart project (package) to be attached.

Using Integration with the Pub Tool

On this page:

- Introduction
- Managing Dart dependencies
- Compiling Dart code into JavaScript using the Pub Serve tool
- Building a Dart application

Introduction

PhpStorm provides interface for using the Pub tool . Alternatively, you can use the tool in the command line mode from the embedded local terminal . The pub tool is intended for installing, uninstalling, and upgrading Dart packages for your project, compiling Dart code into JavaScript for further running or debugging, and for building Dart applications.

The pub tool is included in the Dart SDK, so when you download the SDK you automatically get the pub tool. For details, see Preparing for Dart Development and Dart Download Page .

You can also use the pub tool in the command line mode from the embedded local terminal .

Execution of the pub tool commands is logged in the Messages Tool Window . In this tool window, you can examine the tool log and re-run the last executed command by clicking ▶ on the toolbar.

Managing Dart dependencies

PhpStorm provides interface for running the pub get and pub upgrade actions from the editor. As soon as you open a pubspec.yaml file in the editor, PhpStorm displays a Pub actions pane at the top of the `pubspec.yaml` editor tab. You can use the links on this pane or use the context menu of the editor or the `pubspec.yaml` file in the Project tool window.

The pub tool saves the downloaded packages in caches, creates a `packages` folder next to the `pubspec.yaml` file which contains information on packages and their dependencies. In the `web` folder, the pub tool creates a copy of the `packages` folder. As soon as a subfolder if created under the `web` folder, a copy of the `packages` folder is created in the new subfolder.

All the copies of the `packages` folder are automatically excluded from the project, that is, they are not indexed. PhpStorm excludes these copies on opening a project and every time the pub get or pub upgrade action is invoked. All the excluded copies of the `packages` folder are listed in the exclusion list on the Directories page. You can remove a copy from this list but the copy will be restored during the first automatic exclusion.

To download or upgrade dependencies:

1. Open the `pubspec.yaml` file in the editor.
2. To detect and download dependencies, do one of the following:
   - Click Get Dependencies on the Pub actions pane.
   - Choose Pub: Get Dependencies on the context menu of the editor.
   - In the Project tool window, select the `pubspec.yaml` file and choose Pub: Get Dependencies on the context menu of the selection.
3. To download the up-to-date versions of the existing dependencies, do one of the following:
   - Click Upgrade Dependencies on the Pub actions pane.
   - Choose Pub: Upgrade Dependencies on the context menu of the editor.
   - In the Project tool window, select the `pubspec.yaml` file and choose Pub: Upgrade Dependencies on the context menu of the selection.

If for some reasons the previously downloaded and stored in a cache dependencies are broken, you can clear the cache and download the dependencies anew:

1. Open the `pubspec.yaml` file in the editor.
2. Click Repair Cache on the Pub actions pane.

Compiling Dart code into JavaScript using the Pub Serve tool

When running and debugging Dart web applications, PhpStorm uses the Pub Serve tool. Integration with it is established through the built-in server which acts transparently as a proxy server between your application and the Pub Serve tool. In other words, integration with the Pub Serve tool works invisibly and you do not need to care about it. The tool is invoked automatically when you run or debug a Dart web application, either by opening an HTML file or by launching a run/debug configuration. The tool analyzes the `<script></script>` element in the HTML file. If the element references a Dart file, Pub Serve invokes the Dart2js compiler whereupon the compiled JavaScript files are passed to the browser where they are processed.

The work of the Pub Serve tool is logged in the dedicated Pub Serve tool window which opens when you start running or debugging a Dart web application for the first time during the current PhpStorm session. You can stop the tool by clicking ■ on the toolbar. As soon as you start running or debugging again, the tool restarts automatically.

See Running and Debugging Dart Web Applications for details.

Building a Dart application

When you are ready to deploy your application, you need to build it, which includes generating assets for the project and for all its dependencies and compiling Dart code into JavaScript. Compilation is necessary because

most browsers that work with JavaScript code do not natively process Dart code. The only exception is Dartium . This dart-oriented version of Chromium contains a built-in Dart VM and therefore can run Dart code directly. To be executed in any other browser, Dart code has to be compiled into JavaScript.

With the pub tool, the compilation is performed through the pub serve action using the built-in dart2js compiler.

PhpStorm provides interface for using the pub tool which builds applications through the pub build action. The pub build action compiles the Dart code into JavaScript using the dart2js , generates assets , and stores all the output in the automatically created `build` folder. The folder is marked as excluded, see Configuring Folders Within a Content Root .

See Assets and Transformers for details.

To build a Dart application:

1. Open the `pubspec.yaml` file in the editor or switch to the tab where it is opened.
2. On the pub pane at the top of the tab, click Build.
3. In the dialog box that opens, choose the build mode which determines the behaviour of the dart2js tool:
   - Release: choose this option to have dart2js minify the generated JavaScript code. The term minification or compression in the context of JavaScript means removing all unnecessary characters, such as spaces, new lines, comments without affecting the functionality of the source code. In the Release mode, the source `.dart` files are not included in the build output. This mode is applicable when you are going to publish your application.
   - Debug: choose this option if you do not want to compress the generated JavaScript. In this mode, the generated JavaScript files are included in the build output without compression, as well as the source `.dart` files.
   - Other: choose this option if you want to use a custom transformer and specify the path to it.

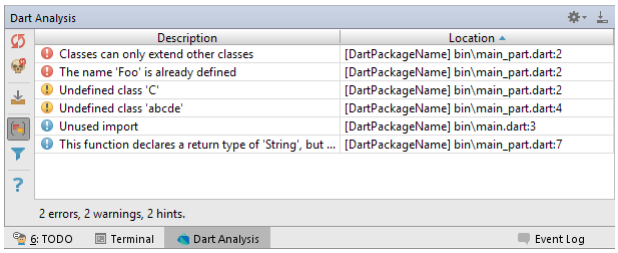
Using Integration with the Dart Analysis Server

On this page:

- Introduction
- Messages list
- Grouping and sorting
- Navigation and speed search
- Configuring the scope of files to view problem reports from

Introduction

PhpStorm provides interface for using the Dart Analyzer tool . The tool is intended for static analysis of your Dart source code. All inconsistencies and potential problems are reported in the dedicated Dart Analysis tool window with the possibility to navigate to the fragment of the source code where the problem was detected.



Messages list

All the problem reports are displayed in a table which consists of two columns, Description and Location, and a summary below.

- The Description column shows the message itself, which can be an error message, a warning, or a hint.
- The Location column shows the following data:
  - The name of the Dart package in which the problem arose. The Dart package name corresponds to the `project name` from `pubspec.yaml` .
  - The relative from the the Dart package root to the file where the problem arose.
  - The line number where the problem arose.
- The Summary area below the table shows the number of detected errors, warnings, or hints and the filter status.

Grouping and sorting

The messages in the Dart Analysis tool window can be grouped and sorted. By default, the messages are primarily sorted by their severity, that is, the error messages are shown at the top of the list, then come warnings, and finally hints are displayed. In each severity group, messages are grouped by the Dart package name, which is super handy for projects with multiple `pubspec.yaml` files. In each package group, the messages are grouped by their file paths. Finally, in each file group, the problems are sorted by the line number where they occurred.

You can re-configure the secondary sorting:

- Click the Location column header to switch between the ascending and descending order of the Dart package name + file path sorting. Note that this does not affect the order of severity groups, that is, errors are always shown first. Line numbers are not taken into consideration either, for each particular file problems are always shown in ascending order of line numbers.
- Click the Description column header to perform the secondary sorting alphabetically by the problem description. Note that this does no affect the primary grouping by severity.

To disable the primary grouping by severity, release the Group by Severity toggle button on the toolbar. After that the severity of a problem is not taken into consideration at all and problem reports are sorted only by their Description or Location (Dart package name + file path). This sorting method guarantees that all the problems for each particular file are grouped together in the table, regardless of their severity.

Navigation and speed search

PhpStorm provides speed search through the messages in the Dart Analysis tol window with the possibility to navigate from a problem report to the fragment of the source code where the problem was detected.

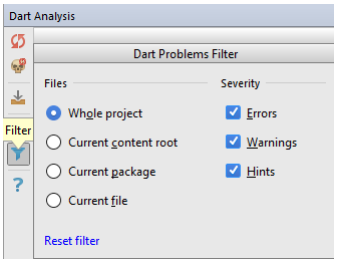
- To locate problems for a specific file or to find a particular message, move the focus to the table and start typing the file name or the text of the message. PhpStorm detects the matching row and selects it.
  Fuzzy search with partial and middle matching is also supported. For example, if you start typing `ain` , the messages related to the `main.dart` fill be also detected.
- To navigate from a problem report to the source code where the problem occurred, select the message in question and choose Jump to Source on the context menu of the selection.
- To have PhpStorm automatically open the source file with the problem reported in the selected row, press the Autoscroll to Source toggle button on the toolbar.

Configuring the scope of files to view problem reports from

You can configure the criteria according to which a problem report is displayed in the tool window or not:

1. Click the Filter button on the toolbar.
2. In the Dart Problems Filter pop-up window, specify the following:
   - In the Severity area, specify the types of messages to be shown, the available types are Errors, Warnings, and Hints. To have the problems of a severity level displayed, select the check box next to this severity level.
   - In the Files area, choose the scope for which you want to see problem reports. The available options are:
     - Whole project
     - Current content root
     - Current package: when this option is selected but no `pubspec.yaml` file is detected up the folder hierarchy starting from the current file, then filtering is performed according to the content root where current file is located.
     - Current file

- Click the Reset all filters link to restore the default filter settings.



Filtering is applied immediately as soon as you change the current settings.

Running and Debugging Dart Applications

Dart is intended for developing Web and command-line applications. PhpStorm supports both workflows.

- For running and debugging Dart command-line applications, PhpStorm provides a dedicated run/debug configuration of the type Dart Command Line Application.
- To run and debug a Dart Web application in a browser, you either need to use the Dartium browser or have the Dart code translated into JavaScript.
  Most browsers that work with JavaScript code do not natively process Dart code. The only exception is Dartium . This dart-oriented version of Chromium contains a built-in Dart VM and therefore can run Dart code directly. To be executed in any other browser, Dart code has to be compiled into JavaScript.

  With the pub tool, the compilation is performed through the pub serve action using the built-in dart2js compiler.

  You can run Dart Web application in any browser, while debugging is supported only in Dartium and Chrome. To run a Dart Web application you just need to open the starling HTML page in the browser of your choice. Debugging a Dart Web application is initiated through a run configuration of the type JavaScript Debug.

In this section:

- Running and Debugging Dart Command-Line Applications
- Running and Debugging Dart Web Applications

Running and Debugging Dart Command-Line Applications

For running and debugging Dart command-line applications, PhpStorm provides a dedicated run/debug configuration of the type Dart Command Line Application.

On this page:

- Running a Dart command-line application
- Debugging a Dart command-line application

Running a Dart command-line application

1. Create a run configuration:
   1. On the main menu, choose Run | Edit Configurations.
   2. Click the Add New Configuration toolbar button, and choose Dart Command Line Application on the context menu.
   3. In the Run/Debug Configuration: Dart Command Line Application dialog box that opens, specify the Dart file to start the application from. The file must contain a `main()` method. Optionally specify the options to launch the Dart Virtual Machine with and the program options to start the application with.
   4. Click OK to save the configuration settings.
2. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Run toolbar button.
3. View the results in the Console tab of the Run tool window.

Debugging a Dart command-line application

With PhpStorm, you can debug Dart command-line applications in two modes:

- Local debugging. In this mode, you launch the application to debug and start a debugging session through one same run configuration.
- Remote debugging. In this mode, you initiate a debugging session and connect to an already running application. This application can be launched on another host machine or you can previously start it on your machine through another run configuration.

To debug a Dart command-line application locally:

1. Configure and set breakpoints in the Dart code.
2. Create a run configuration:
   1. On the main menu, choose Run | Edit Configurations.
   2. Click the Add New Configuration toolbar button, and choose Dart Command Line Application on the context menu.
   3. In the Run/Debug Configuration: Dart Command Line Application dialog box that opens, specify the Dart file to start the application from. The file must contain a `main()` method. Optionally specify the options to launch the Dart Virtual Machine with and the program options to start the application with.
   4. Click OK to save the configuration settings.
3. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button.
4. In the Debug Tool Window that opens, step through the program , stop and resume program execution, examine it when suspended , etc.

To debug a Dart command-line application remotely:

1. Create a Dart Remote Debug run configuration:
   1. On the main menu, choose Run | Edit Configurations.
   2. Click the Add New Configuration toolbar button, and choose Dart Command Line Application on the context menu.
   3. In the Run/Debug Configuration: Dart Remote Debug dialog box that opens, specify the host where the Dart application is running and the port for the debugger to communicate with this host. The specified port is shown in the Command line arguments for the remote Dart VM read-only field. Note that the application must be started exactly with these arguments, so do not forget to provide this information to the one who will start it on the remote host or specify the arguments in the Dart Remote Debug run configuration if you are going to launch the application yourself on the localhost.
   4. From the Search Sources in drop-down list, choose the Dart project to debug if your PhpStorm project contains several Dart projects configured as content roots.
   5. Click OK to save the configuration settings.
2. Make sure the command-line application to debug is running with the arguments from the Command line arguments for the remote Dart VM read-only field in the Dart Remote Debug run configuration.
3. Choose the newly created Dart Remote Debug configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button.
4. In the Debug Tool Window that opens, step through the program , stop and resume program execution, examine it when suspended , etc.

Running and Debugging Dart Web Applications

On this page:

- Introduction
- Running a Dart web application
- Debugging a Dart web application

Introduction

To run and debug a Dart Web application in a browser, you either need to use the Dartium browser or have the Dart code translated into JavaScript.

most browsers that work with JavaScript code do not natively process Dart code. The only exception is Dartium . This dart-oriented version of Chromium contains a built-in Dart VM and therefore can run Dart code directly. To be executed in any other browser, Dart code has to be compiled into JavaScript.

With the pub tool, the compilation is performed through the pub serve action using the built-in dart2js compiler.

You can run Dart Web application in any browser, while debugging is supported only in Dartium and Chrome. To run a Dart Web application you just need to open the starling HTML page in the browser of your choice. Debugging a Dart Web application is initiated through a run configuration of the type JavaScript Debug.

When running and debugging Dart web applications, PhpStorm uses the Pub Serve tool. Integration with it is established through the built-in server which acts transparently as a proxy server between your application and the Pub Serve tool. In other words, integration with the Pub Serve tool works invisibly and you do not need to care about it. The tool is invoked automatically when you run or debug a Dart web application, either by opening an HTML file or by launching a run/debug configuration. The tool analyzes the `<script></script>` element in the HTML file. If the element references a Dart file, Pub Serve invokes the Dart2js compiler whereupon the compiled JavaScript files are passed to the browser where they are processed.

The work of the Pub Serve tool is logged in the dedicated Pub Serve tool window which opens when you start running or debugging a Dart web application for the first time during the current PhpStorm session. You can stop the tool by clicking ■ on the toolbar. As soon as you start running or debugging again, the tool restarts automatically.

Running a Dart web application

No run configuration is required for launching Dart Web applications.

1. In the editor, open the HTML file with the Dart reference. This HTML file does not necessarily have to be the one that implements the starting page of the application.
2. Do one of the following:
   - Choose View | Preview file in on the main menu or press `Alt+F2`. Then select the desired browser from the pop-up menu.
   - Hover your mouse pointer over the code to show the browser icons bar: Click the icon that indicates the desired browser.

   If you choose Dartium which has a built-in Dart virtual machine, simply the Dart code is executed. If you choose another browser, the Dart code is compiled into JavaScript through the Dart2js tool, which is a part of the Pub tool. Whether such compilation is required or not is decided by the Pub Serve tool.

Debugging a Dart web application

Debugging of Dart Web applications is supported only in Dartium and Chrome through the JetBrains Chrome Extension. A debugging session is initiated through a run configuration of the type JavaScript Debug.

1. Install and configure the Chrome Extension in your browser:
   1. Open the Chrome Web Store at https://chrome.google.com/webstore/detail/jetbrains-ide-support/hmhgeddbohgjknpmjagkdomcpobmllji .
   2. On the JetBrains IDE Support page that opens, click the Add to Chrome button + FREE . Then click Add in the Confirm New Extension dialog box that opens. Learn more at Using JetBrains Chrome Extension , chapter Installing the JetBrains Chrome Extension.
   3. Make sure the extension connects to PhpStorm through the same port as the one specified in the Built-in server port field on the Debugger page of the Settings dialog box. To do that, right-click the JB icon, choose Options on the context menu, and then check the value in the Port spin box. Learn more at Using JetBrains Chrome Extension , chapter Changing the Port for Connecting to PhpStorm.
2. Configure and set breakpoints in the Dart code.
3. Create a run/debug configuration of the type JavaScript Debug :
   1. On the main menu, choose Run | Edit Configurations.
   2. Click the Add New Configuration toolbar button +, and choose JavaScript Debug on the context menu.
   3. In the Run/Debug Configuration:JavaScript Debug dialog box that opens, choose the browser to debug the application in. If you choose Dartium which has a built-in Dart virtual machine, simply the Dart code is executed. If you choose Chrome, the Dart code is compiled into JavaScript through the Dart2js tool, which is a part of the Pub tool. Whether such compilation is required or not is decided by the Pub Serve tool.
      Specify the URL address of the HTML file the references the Dart code to debug. Make sure that the port specified in this URL address is the same as the Built-in server port on the Debugger page of the Settings dialog box and the port specified in the settings for the Chrome Extension.
4. Initiate a debugging session: choose the created run configuration from the Edit configurations drop-down list on the toolbar and click . PhpStorm opens the specified URL in the chosen browser and opens the Debug Tool Window .
5. In the Debug tool window, step through the program , stop and resume the program, examine it when suspended, etc. See Stepping Through the Program , Pausing and Resuming the Debugger Session , and Examining Suspended Program for details. ,

Testing Dart

On this page:

- Before you start
- Creating a Dart test run configuration

Before you start

1. Make sure the Dart development environment is configured, see Preparing for Dart Development .
2. Make sure the Dart plugin is activated. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Creating a Dart test run configuration

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button +, and choose Dart Test on the context menu.
3. In the Run/Debug Configuration: Dart Test dialog that opens, specify the name of the configuration and the scope of the tests to execute. The available options are:
   - All in File: choose this option to to have PhpStorm run all the test methods in a file and specify this file in the Test File text box.
   - All in Folder: choose this option to to have PhpStorm run all the test methods from the files stored in a folder and specify this folder in the Test Folder text box.
   - Group or test by name: choose this option to to have PhpStorm run a specific tast or a group of tests.
     - In the Test File text box, specify the file where the test or the group is defined.
     - In the Group or Test Name text box, type the name of the test or the group of tests to execute.

     See more about grouping tests at https://www.dartlang.org/articles/dart-unit-tests/#grouping-tests .

Databases and SQL

In this section:

- Databases and SQL
  - Databases and SQL support
- Managing Data Sources
- Working with the Database Tool Window
- Working with Database Consoles
- Working with the Table Editor
- Running SQL Script Files
- Running Injected SQL Statements

Databases and SQL support

PhpStorm features for working with databases and SQL include:

- Integration with the most popular database management systems such as Oracle , PostgreSQL , MySQL , SQL Server and others. To be able to work with your databases, you should define them as data sources. See Managing Data Sources .
- Database tool window for managing data structures in your databases (View | Tool Windows | Database). See Working with the Database Tool Window .
- Database consoles that let you compose and execute SQL statements as well as analyze and modify retrieved data (`Ctrl+Shift+F10` in the Database tool window). See Working with Database Consoles .
- Table editor that provides a GUI for working with table data (`F4` in the Database tool window). See Working with the Table Editor .
- SQL code generation and editing features in the database consoles and the editor, e.g.
  - Predefined code snippets (a.k.a. live templates) such as for `CREATE TABLE` , `SELECT` , `INSERT` , `UPDATE` and other statements (`Ctrl+J`).
  - Auto-completion and highlighting of SQL keywords, and table and column names.
  - Data type prompts for columns (`Ctrl+P`).

  Standardized and DBMS-specific SQL dialects are supported.
- Structure view for tables in the table editor and Database Console tool window as well as for open database consoles and SQL files (`Ctrl+F12`). See e.g. Using the Structure view to sort data, and hide and show columns.
- Quick documentation view for database objects and table cells (`Ctrl+Q`). See e.g. Using the quick documentation view.
- Navigation capabilities, e.g.
  - From a table or column reference to its definition: `Ctrl+B` .
  - To the view of a table or column in the Database tool window: `Alt+F1` | Database View.
  - By means of the navigation bar: `Alt+Home` .
  - By means of the Switcher: `Ctrl+Tab` .
- Database diagrams (`Ctrl+Alt+U` or `Ctrl+Shift+Alt+U` in the Database tool window).

Managing Data Sources

On this page:

- Data sources
- Defining a database as a data source
- Creating a DB data source for H2 or SQLite by means of drag and drop
- Creating DB data sources by importing connection settings
- Creating a DDL data source
- Creating a DDL data source by means of drag and drop
- Changing data source settings
- Removing data sources

Data sources

To be able to work with your databases in PhpStorm, you should define them as data sources.

In addition to data sources that correspond to real databases (DB data sources), PhpStorm also supports DDL data sources. These are represented by one or more SQL files containing data definition language statements (SQL DDL statements).

Metaphorically, DDL data sources function as databases without data.

Data sources provide the basis for SQL coding assistance and code validation.

Defining a database as a data source

1. Open the Database tool window and click on the toolbar.
2. In the Data Sources and Drivers dialog that opens, click + and select the database management system (DBMS) that you are using.
3. In the Name field, if necessary, edit the name of the data source.
4. If there is the message Download missing driver files in the lower part of the dialog, specify the driver files.
   (To interact with a database, PhpStorm needs a database driver. The drivers, generally, are DBMS-specific.)

   Do one of the following:
   - To download the necessary driver, click the Download link.
   - To specify the driver files that you already have available on your computer, click the <DriverName> link to the right of Driver.
     On the page where the driver settings are shown, under Driver files, click + and select the files in the dialog that opens.

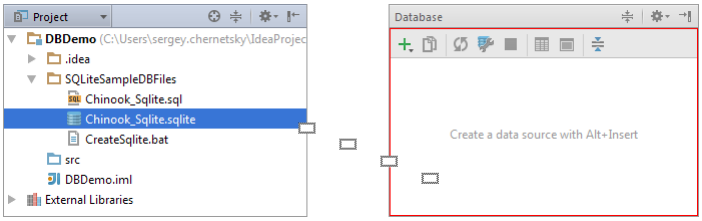
     Go back to the page with the data source settings.
5. Specify the database settings.
   For most of the DB management systems, these are the database host name (or IP address), port, the database name, and also your database user name and password.

   If your database is a local file or directory (which may be the case for SQLite, H2, Derby or HSQLDB), you should specify the location of that file or directory. To do that, click ... to the right of the File or Path field and select the database file or directory in the dialog that opens.
6. To access your database using SSH or SSL, select the SSH/SSL tab and specify the corresponding settings there.
7. To make sure that the settings are correct and PhpStorm can properly interact with your database, click Test Connection on the General tab.
8. If the settings you have specified identify more than one database or schema, you can select the database or schema you are going to work with. To do that, use the Schemas tab .
9. Click OK to save the settings and close the dialog.
   As a result, your new data source is shown in the Database tool window. The input pane of the associated database console opens to the right of the tool window.

Creating a DB data source for H2 or SQLite by means of drag and drop

If you have H2 or SQLite database files available locally, you can create DB data sources for them by dragging the files to the Database tool window. The files can be dragged from the Project tool window, or from your file manager (e.g. Explorer or Finder).

1. If the database files are in your project folder, open the Project tool window. Otherwise, open your file manager.
2. Open the Database tool window .
3. Select the file or files of interest in the Project tool window, or in your file manager.
4. Drag the selected file or files into the Database tool window. (For each of the files a separate data source will be created.)



5. If you don't have the necessary database driver files yet, you can download them now. Click on the toolbar of the Database tool window. (Alternatively, select Properties from the context menu.)
6. In the Data Sources and Drivers dialog that opens, within the line Driver files missing, click the Download link.
7. Click Test Connection to make sure that PhpStorm can properly communicate with the database.
8. Click OK in the Data Sources and Drivers dialog.

Creating DB data sources by importing connection settings

Files that contain database connection settings (`configuration.php`) can be used for creating DB data sources.

1. If the files that you want to import the settings from are not in your project yet, copy them there.
2. Open the Database tool window .
3. Do one of the following:
   - Click + on the toolbar and select Import from sources.
   - Right-click the area under the toolbar or any of the existing data sources, point to New and click Import from sources.

   The Data Sources and Drivers dialog opens. The names of candidate data sources are shown in the left-hand pane in green.
4. Specify the driver files if they are missing.
   Do one of the following:
   - To download the necessary driver, click the Download link.
   - To specify the driver files that you already have available on your computer, click the <DriverName> link to the right of Driver.
     On the page where the driver settings are shown, under Driver files, click + and select the files in the dialog that opens.

     Go back to the page with the data source settings.
5. Click Test Connection to make sure that PhpStorm can properly communicate with the database.
6. Click OK in the Data Sources and Drivers dialog.

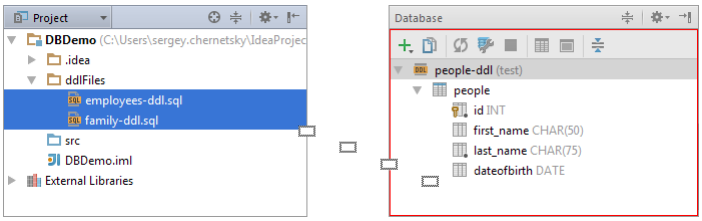
Creating a DDL data source

1. Open the Database tool window and click on the toolbar.
2. In the Data Sources and Drivers dialog that opens, click + and select DDL Data Source.
3. In the Name field, if necessary, edit the name of the data source.
4. Under DDL Files, click + and select the necessary SQL file or files in the dialog that opens.
5. From the Extend list, if necessary, select another data source as a parent. As a result, the data source whose properties you are editing will "inherit" all the DDL definitions from its parent.
6. Click OK to save the settings and close the dialog.

Creating a DDL data source by means of drag and drop

You can create DDL data sources by dragging DDL SQL files to the Database tool window. The files can be dragged from the Project tool window, or from your file manager (e.g. Explorer or Finder).

1. If the necessary DDL SQL files are in your project folder, open the Project tool window. Otherwise, open your file manager.
2. Open the Database tool window .
3. If you already have one or more DDL data sources defined but there are no DB data sources, expand one of the data source nodes in the Database tool window.
4. Select the file or files of interest in the Project tool window, or in your file manager.
5. Drag the selected file or files into the Database tool window. For a new data source to be created, the red border, when dropping the file or files, should surround most of the window area (rather than one of the existing data sources).



Changing data source settings

1. Open the Database tool window and select the data source of interest.
2. Do one of the following:
   - Click on the toolbar.
   - Select Properties from the context menu.
   - Press `Alt+Enter` .
3. In the Data Sources and Drivers dialog that opens, edit the settings as necessary. See:
   - DB data source settings
   - DDL data source settings

Removing data sources

To remove unnecessary data sources, you can use the Database tool window or the Data Sources and Drivers dialog.

Using the Database tool window. Select the data sources to be removed and do one of the following:

- Press `Delete`.
- Select Delete from the context menu.
- Select Edit | Delete.

Using the Data Sources and Drivers dialog. Select the data sources to be removed and do one of the following:

- Click – on the toolbar.
- Press `Delete`.
- Select Remove from the context menu.

Working with the Database tool window

The Database tool window provides access to functions for working with databases and DDL data sources. It lets you view and modify data structures in your databases, and perform other associated tasks.

- Opening the Database tool window
- Adjusting the view
- Creating a data source
- Synchronizing the view of a DB data source
- Finding items
- Finding occurrences of an item name
- Creating a copy of a data source
- Creating a table, a column, an index, or a primary or foreign key
- Viewing basic info about an item
- Renaming a table or column
- Modifying the definition of a table, column, index or a foreign key
- Previewing changes
- Removing items
- Opening a default database console
- Creating and opening a new database console
- Opening the table editor
- Generating DDL definitions for data sources, tables and columns
- Comparing table structures
- Saving table data in a file
- Viewing diagrams
- Closing database connections

See also, Database Tool Window .

Opening the Database tool window

Do one of the following:

- Select View | Tool Windows | Database.
- Point to or in lower-left corner of the workspace, and then click Database.
- Click Database on the right-hand tool window bar (if the tool window bars are currently shown).

Adjusting the view

You can adjust the view in the tool window by turning the corresponding view options on and off. To access those options, click on the title bar.

For more information, see View options.

Creating a data source

To start creating a data source , you can use the New command when the Database tool window is active, e.g.:

- File | New
- on the toolbar
- New in the context menu
- `Alt+Insert`

The DDL Data Source option is for creating a DDL data source. Other data source options correspond to different scenarios of creating a DB data source:

- Data Source. A "usual way" of creating a data source. In this scenario, you start by selecting your DBMS.
- Data Source from URL. In this scenario, you start by specifying your database URL.
- Data Source from Path. In this scenario, you start by specifying your database location (a local file or folder). This option is appropriate only for Derby, H2, HSQLDB and SQLite.
- Import from sources. If you have files that contain database connection settings, you can create data sources by importing those settings. See Creating DB data sources by importing connection settings.

You can also start creating a data source in the Data Sources and Drivers dialog . Open the dialog (e.g.) and use the Add command there: Add from the context menu, on the toolbar, or `Alt+Insert`.

For more information, see Managing Data Sources .

Synchronizing the view of a DB data source

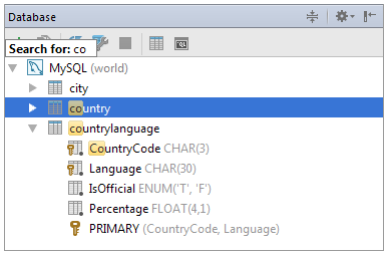
If the Auto sync option for a DB data source if off, the only way to synchronize its view in the Database tool window with the actual state of the database is by using the Synchronize command.

1. Select the item whose view you want to synchronize. This may be a DB data source, schema or table.
2. Do one of the following:
   - Press `Ctrl+Alt+Y`.
   - Click on the toolbar.
   - Select Synchronize from the context menu.

Finding items

To find an item of interest, simply start typing its name. The specified text within item names is highlighted, and the first of the items that contains the specified text is

selected.



Finding occurrences of an item name

To find occurrences of the name of an item (e.g. table or column) in your files and consoles:

1. Select the item of interest.
2. Do one of the following:
   - Press `Alt+F7`.
   - Select Find Usages from the context menu.
   - Select Edit | Find | Find Usages in the main menu.

Creating a copy of a data source

1. Select the data source of interest.
2. Do one of the following:
   - Click on the toolbar.
   - Select Duplicate from the context menu.
   - Press `Ctrl+D`.

Creating a table, a column, an index, or a primary or foreign key

1. Depending on what you are going to create:
   - To create a table, select a schema, table or column within the target DB data source.
   - To create a column, select the target table or a column within that table.
   - To create an index, or a primary or foreign key, select the column or columns for which you want to create an index, or a primary or foreign key constraint.
2. Carry out the New command and select the item to be created. E.g. for a table, do one of the following:
   - Select File | New | Table.
   - Click + and select Table.
   - In the context menu, select New | Table.
   - Press `Alt+Insert` and select Table.
3. In the dialog that opens , specify the item definition.

Viewing basic info about an item

You can view basic info about an item in the quick documentation view. For a table, for example, the first ten rows and the table definition (the `CREATE TABLE` statement) are shown.

To open the quick documentation view, select the item of interest and do one of the following:

- Select View | Quick Documentation.
- Press `Ctrl+Q`.

See also, Allow database access .

Renaming a table or column

1. Select the table or column to be renamed.
2. Do one of the following:
   - Select Refactor | Rename.
   - Select Rename from the context menu.
   - Press `Shift+F6`.
3. Use the dialog that opens to specify a new name and associated options.

Modifying the definition of a table, column, index or a foreign key

1. Select the item whose definition you want to change. This may be a table, a column, an index or a foreign key.
2. Do one of the following:
   - Select Modify <item_type> from the context menu (e.g. Modify Table).
   - Press `Ctrl+F6`.
3. Use the dialog that opens to change the item definition.

Previewing changes

Changes to database objects sometimes assume associated changes to SQL script files and statements in database consoles. For example, you may be changing the name of a table, and this name may be used in your files and consoles.

In such cases, you can look at potential changes, and decide where those changes are desirable and where aren't.

Potentially affected code fragments are shown in the Find tool window when you click Preview in the corresponding dialogs. Here is an overview of some of the available controls:

- Exclude (`Delete`) and Remove (`Alt+Delete`). Use these context menu commands for the items that shouldn't be changed.
- Execute SQL Script. If this option is on, and you click Do Refactor, the corresponding SQL statements are run to modify the corresponding database objects.
- Open in Console. Use this button to open the corresponding SQL statements in a database console .
- Do Refactor. Click this button to change the corresponding code fragments and, if the Execute SQL Script option is on, to run the corresponding SQL statements.

Removing items

To remove data sources, use the Remove command (Edit | Remove, Remove from the context menu, or `Delete` on the keyboard).

To remove all other item types such as tables, columns and indexes, use the Drop command (Edit | Drop, Drop from the context menu, or `Delete` on the keyboard).

For removing primary key constraints, in addition, there is the Drop Associated | Primary Key context menu command.

See also, Confirm Drop dialog .

Opening a default database console

1. Select the DB data source of interest or any node within it.
2. Do one of the following:
   - Click on the toolbar.
   - Select Open Console from the context menu.
   - Press `Ctrl+Shift+F10` .

For more information, see Working with Database Consoles .

Creating and opening a new database console

1. Select the DB data source of interest or any node within it.
2. Do one of the following:
   - Select Open New Console from the context menu.
   - Click + and select Console File.

For more information, see Working with Database Consoles .

Opening the table editor

1. Select the table of interest.
2. Do one of the following:
   - Click on the toolbar.
   - Select Table Editor from the context menu.
   - Press `F4` .

For more information, see Working with the Table Editor .

Generating DDL definitions for data sources, tables and columns

1. Select the data source, table or column of interest.
2. Do one of the following:
   - Select Copy DDL from the context menu.
   - Press `Ctrl+Shift+C` .

   The definitions are generated on the clipboard and you can paste them into a database console or an SQL file.

Comparing table structures

1. Select two data sources, schemas or tables.
2. Do one of the following:
   - Select Compare from the context menu.
   - Press `Ctrl+D` .

The comparison results are shown in the differences viewer .

Saving table data in a file

1. Right-click the table or view of interest.
2. Point to Dump Data To File and select the output format (e.g. Comma Separated Values (CSV)).
3. In the dialog that opens, specify the file location and name.

Viewing diagrams

To open a diagram for a data source, schema or table, select the item of interest and do one of the following:

- Press `Ctrl+Shift+Alt+U` or `Ctrl+Alt+U` .
- In the context menu, select Diagrams, and then select Show Visualisation or Show Visualisation Popup.

Closing database connections

PhpStorm connects to databases automatically, when needed. (The names of the data sources with open database connections are shown in the Database tool window in bold.)

To close unnecessary database connections, select the corresponding data sources and do one of the following:

- Click ■ on the toolbar.
- Select Disconnect from the context menu.
- Press `Ctrl+F2`.

Working with Database Consoles

Database consoles let you compose and execute SQL statements for databases defined in PhpStorm as data sources. They also let you analyze and modify the retrieved data.

The following standardized and DBMS vendor-specific SQL dialects are supported: DB2, Derby, H2, HSQLDB, MySQL, Oracle, Oracle SQL*Plus, PostgreSQL, SQL Server, SQL92, SQLite, and Sybase .

One database console is created for a data source automatically when a data source is created. If necessary, you can create additional consoles.

Database consoles are persistent: they are stored as SQL files.

A database console created in one of your projects can then be accessed from any other project.

General console usage instructions are on this page:

- Creating a database console
- Opening a database console
- Viewing and modifying console settings
- Changing the SQL dialect
- Closing a console
- Managing database consoles

More specific subjects are discussed in:

- Writing and Executing SQL Statements
- Working with Query Results

See also, Database Console .

Creating a database console

When you create a DB data source, one database console for that data source is created automatically. If necessary, you can create additional consoles.

To create a database console, you can use the Database tool window or the Scratches view of the Project tool window . The procedure is the same in both cases:

Select the data source of interest or any node within it, and do one of the following:

- Select File | New | Console File from the main menu.
- Select New | Console File from the context menu
- Press `Alt+Insert` and select Console File.

In the Database tool window, you can also use + | Console File and the Open New Console context menu command.

Opening a database console

The Database tool window lets you open only default consoles, i.e. the ones that were created by PhpStorm automatically.

To open the consoles that you created yourself, you should use the Scratches view of the Project tool window .

Using the Database tool window. Select the data source of interest or any node within and do one of the following:

- Click on the title bar if the toolbar is hidden.
- Click on the toolbar if the toolbar is shown.
- Select Open Console from the context menu.
- Press `Ctrl+Shift+F10`.

Using the Scratches view. For the console of interest, do one of the following:

- Double-click the console.
- Select View | Jump to Source from the main menu.
- Select Jump to Source from the context menu.
- Press `F4`.

Viewing and modifying console settings

Before actually starting to use a console, you may want to take a look at the console settings and adjust them to your needs.

- To access these settings, click on the toolbar of the input pane or on the toolbar of the Database Console tool window. (Alternatively, `Ctrl+Alt+S` | Tools | Database.)

As a result, the Database page of the Settings / Preferences dialog will open.

Changing the SQL dialect

By default, the SQL dialect used in a console is defined by the DBMS of an associated data source. If for some reason you want to use a different dialect:

- Right-click the editing area of the input pane, select Change Dialect (<CurrentDialect>), and select the necessary dialect.
  In addition to particular dialects, also the following option is available:
  - <Generic>. Basic SQL92-based support is provided including completion and highlighting for SQL keywords, and table and column names. Syntax error highlighting is not available. So all the statements in the input pane are always shown as syntactically correct.

Closing a console

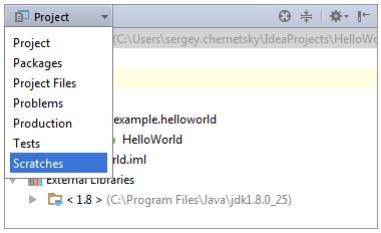
1. Click (`Ctrl+Shift+F4`) to close the Database Console tool window.

2. Click × on the editor tab (`Ctrl+F4`) to close the input pane.

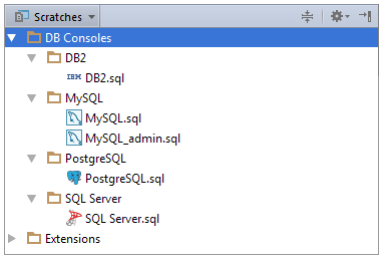
Managing database consoles

To manage your database consoles, use the Scratches view of the Project tool window .

To open this view, select Scratches from the list in the left-hand part of the title bar.



The view shows the existing database consoles (represented by SQL files) grouped by your data sources (shown as folders). The default consoles (the ones that were created by PhpStorm automatically) have the same names as the corresponding data sources.



You can:

- Create new consoles. Select the target data source or a node within it, and do one of the following:
  - Select File | New | Console File from the main menu.
  - Select New | Console File from the context menu
  - Press `Alt+Insert` and select Console File.
- Rename your consoles. When a new console is created, it has the name of the data source with a number at the end, e.g. `MySQL_1` . If you want to give a console a more descriptive name, select the console and do one of the following:
  - Select Refactor | Rename from the main or the context menu.
  - Press `Shift+F6` .

  Then, specify a new name in the dialog that opens.
- Save your console files in arbitrary directories. Select the console and then select Refactor | Copy (`F5`) or File | Save As (`Ctrl+Shift+S`). Specify the file name and location in the dialog that opens.
- Group your consoles. This is done by creating folders and then dragging your consoles into those folders.
- Open your consoles. Select the consoles of interest and do one of the following:
  - Select View | Jump to Source from the main menu.
  - Select Jump to Source from the context menu.
  - Press `F4` .
- View the history of changes for your consoles. Select File | Local History | Show History from the main menu or Local History | Show History from the context menu.
- Delete individual consoles and groups of consoles. Use Edit | Delete, Delete from the context menu or `Delete` on the keyboard.

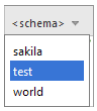
Writing and Executing SQL Statements

Use the input pane of the database console to compose and execute your SQL statements, and also to perform other, associated tasks. (The input pane is shown as a tab in the editor.)

- Selecting the default schema or database
- Composing SQL statements
- Editing data for INSERT statements in table format
- Navigating to a table or column view in the Database tool window
- Configuring the Execute command
- Executing an SQL statement
- Executing parameterized statements
- Executing a group of statements
- Executing all statements
- Executing a part of a statement (e.g. a subquery)
- Executing auto-memorized statements
- Outputting the result of a SELECT statement into a file
- Using the error notification bar
- Canceling running statements
- Managing database transactions
- Showing execution plans
- Showing DBMS_OUTPUT for Oracle

Selecting the default schema or database

You can select the default schema or database by using the list in the right-hand part of the toolbar. If you do so, you'll be able to omit the name of that schema or database in your statements.

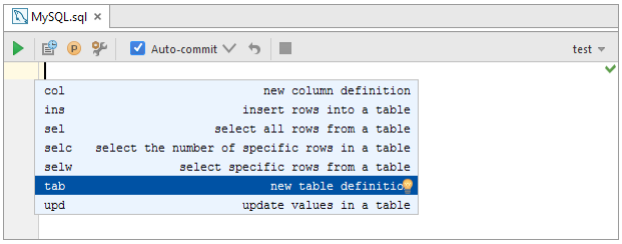


(You cannot switch schemas like this for read-only MySQL data sources, see Read-only .)

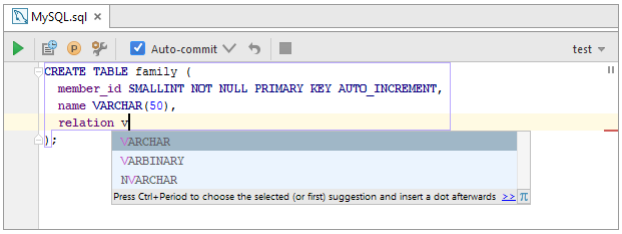
Composing SQL statements

When composing your SQL statements, use:

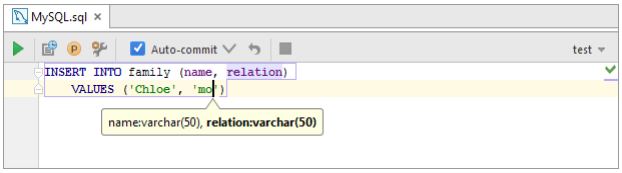
- Predefined patterns (Ctrl+J or Code | Insert Live Template).



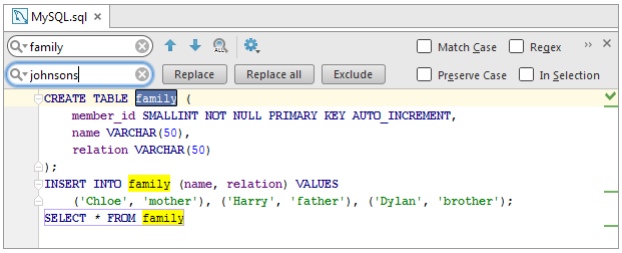
- Auto-completion and highlighting of SQL keywords, and table and column names.



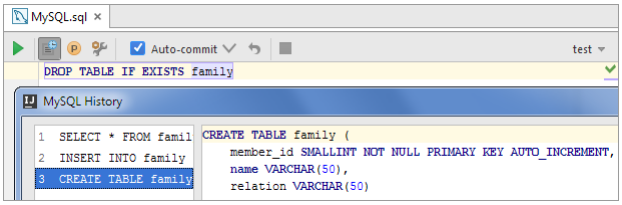
- Data type prompts for columns (Ctrl+P or View | Parameter Info).



- Advanced find and replace capabilities (Ctrl+F or Edit | Find | Find, and Ctrl+R or Edit | Find | Replace).



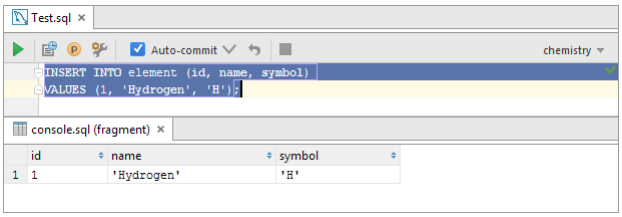
- The console history (or Ctrl+Alt+E). See Executing auto-memorized statements.



See also, Navigating to a table or column view in the Database tool window.

Editing data for INSERT statements in table format

1. Select the `INSERT` statement of interest.
2. Select Edit as Table from the context menu. As a result, the table editor opens.



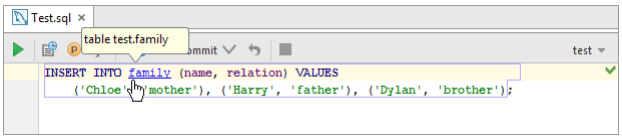
3. Use context menu commands and associated shortcuts for working with the data in the table editor.

Navigating to a table or column view in the Database tool window

When composing a statement, it's sometimes useful to take a look at the structure of a table, or to see the info about a column (field) in the context of the table to which it belongs. For such purposes, PhpStorm provides the ability to switch from the name of a table or column in the input pane to its view in the Database tool window.

The following ways are available for using this feature:

- Place the cursor within the name of the table or column of interest. Then use `Ctrl+B`. (Alternatively, you can use Navigate | Declaration from the main menu or Go To | Declaration from the context menu.)
- Press and hold the `Ctrl` key, and point to the name of interest. When the text turns into a hyperlink, click the hyperlink.



Configuring the Execute command

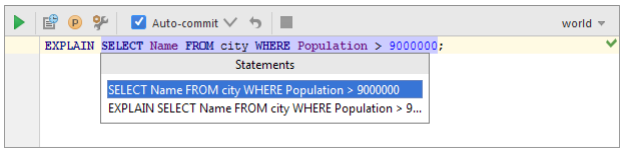
The Execute command (on the toolbar, `Ctrl+Enter` or Execute from the context menu) is used to run your statements.

PhpStorm provides many options for the Execute command depending on the cursor position and on whether there is a selection.

The options are specified on the Tools | Database page in the Settings / Preferences dialog (File | Settings | Tools | Database on Windows and Linux; PhpStorm | Preferences | Tools | Database on OS X). For more information, see Execute in Console.

Executing an SQL statement

1. Place the cursor within the statement.
2. Do one of the following:
   - Click on the toolbar.
   - Press `Ctrl+Enter`.
   - Select Execute from the context menu.
3. Select the statement or statements to be run. (The suggestion list always contains an item for running all the statements.)

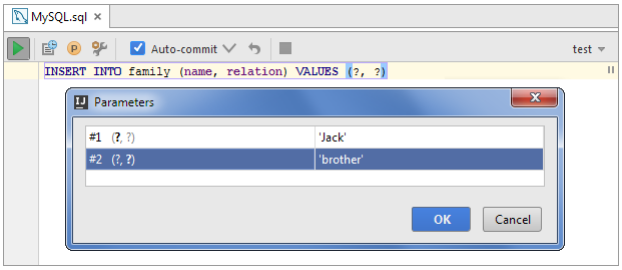


See also, Execute in Console.

Executing parameterized statements

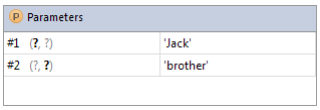
Your statements can contain parameters, however, by the time you execute such statements the values of the parameters must be specified. There are the following ways of specifying the parameter values:

- Click on the toolbar or press `Ctrl+Enter` to execute the statement. In the dialog that opens, specify the parameter values and click OK.



(To start editing a value, switch to the corresponding table cell and start typing. To indicate that you have finished editing a value, press `Enter` or switch to a different cell. To quit the editing mode and restore an initial value, press `Escape`.)

- Alternatively, you can open the Parameters pane in the Database Console tool window (on the toolbar) and specify the corresponding values there. (The values are edited in the same way as in the corresponding dialog.) Then execute the statement (on the toolbar or `Ctrl+Enter`).



For more information, see Parameters pane.

See also, User Parameters and Always review parameters before execution.

Executing a group of statements

To execute a group of statements that follow one another in the console, select (highlight) the statements (to select all the statements, use `Ctrl+A`) and do one of the following:

- Click on the toolbar.
- Press `Ctrl+Enter`.
- Select Execute from the context menu.

See also, Using the error notification bar and Execute in Console.

Executing all statements

To execute all the statements contained in a console, as an alternative to the Execute command, you can use the Run console.sql command.

This command is available in the context menu, and its keyboard equivalent is `Ctrl+Shift+F10`.

The Run console.sql command, generally, runs faster but:

- The statements with parameters don't run.
- Retrieved data for the `SELECT` statements are not shown.

Executing a part of a statement (e.g. a subquery)

To execute a part of a statement (e. g. a subquery), select (highlight) the fragment that you want to execute and do one of the following:

- Click on the toolbar.
- Press `Ctrl+Enter`.
- Select Execute from the context menu.

See also, Execute in Console.

Executing auto-memorized statements

As you run SQL statements in the consoles, PhpStorm memorizes them. So, at a later time, you can view the statements you have already run and, if necessary, run them again.

To open the dialog where the auto-memorized statements are shown (the History dialog), do one of the following:

- Click on the toolbar.
- Press `Ctrl+Alt+E`.

There are two panes in the History dialog. The left-hand pane shows the list of the statements that you have run. For "long" statements, only their beginnings are shown. When you select a statement in this pane, the overall statement is shown in the pane to the right.

You can filter the information: just start typing. As a result, only the statements that contain the typed text will be shown.

You can copy the statements from the History dialog into the input pane of the console. To copy a statement, do one of the following:

- Double-click the statement to be copied.
- Select the statement of interest and press `Enter`.
- Select the statement and click OK.

(Once the statement is in the input pane, you can run it straight away.)

You can delete unnecessary memorized statements. To delete a statement, select the statement in the History dialog and press `Delete`.

Outputting the result of a SELECT statement into a file

Instead of the Result pane of the Database Console tool window, you can output the result of a `SELECT` statement into a file.

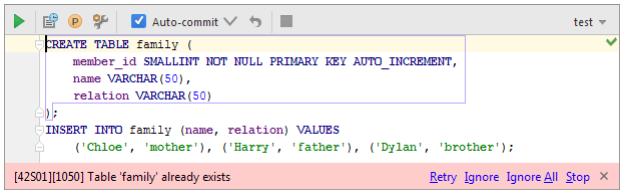
1. Right-click the `SELECT` statement of interest.
2. Point to Execute to File and select the output format.

3. Specify the output file location and name.

Using the error notification bar

If when running a statement an error occurs, an error notification bar appears in the lower part of the input pane.

This bar may be particularly useful when executing a sequence of statements (see Executing a group of statements) because in such a case it lets you select how to react.



The options are:

- Retry. Execute the sequence of statements starting from the one that caused the error.
- Ignore. Skip the erroneous statement and execute the sequence starting from the next statement. If another error occurs, the error notification bar will appear again.
- Ignore All. Skip the erroneous statement and execute the sequence starting from the next statement. If other errors occur, all the erroneous statements will be skipped and the error notification bar won't appear for these statements.
- Stop. Stop the execution of the sequence.

Showing the error notification bar in the input pane is enabled or disabled in the Settings dialog (the Show error notifications in editor check box on the Database page).

Canceling running statements

To terminate execution of the current statement or statements, do one of the following:

- Click on the toolbar of the input pane, or on the toolbar of the Dababase Console tool window.
- Press `Ctrl+F2` .

Managing database transactions

The Auto-commit check box, and the Commit and Rollback icons on the toolbar let you manage database transactions.

The Auto-commit check box is used to turn the autocommit mode for the database connection on or off.

In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back.

If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the Commit ( ) or Rollback ( ) command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements.

In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.

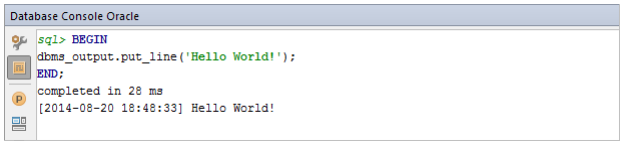
Showing execution plans

The following context menu commands let you show an execution plan (a.k.a. explain plan) for a statement:

- Explain Plan. The result is shown in a mixed tree/table format on a dedicated Plan tab.
- Explain Plan (Raw). The result is shown in table format. (Technically, `EXPLAIN <CURRENT_STATEMENT>` or similar statement is executed.)

Showing DBMS_OUTPUT for Oracle

For Oracle, you can enable or disable showing the contents of the DBMS_OUTPUT buffer in the output pane. To do that, use on the toolbar of the Database Console tool window (`Ctrl+F8`).



Working with Query Results

When you run a query (a `SELECT` statement) in the console, the data retrieved from the database are shown in table format in the Result pane of the Database Console tool window. Depending on the settings, a new Result tab opens for each query, or one and the same tab is used. In the latter case, the results on the tab are updated for each next query.

Use the Result pane to sort, add, edit and remove the data as well as to perform other, associated tasks.

- Hiding or showing the toolbar
- Pinning the Result tab
- Switching between subsets of rows
- Making all rows visible simultaneously
- Navigating to a specified row
- Navigating to related records
- Sorting data
- Reordering columns
- Hiding and showing columns
- Restoring the initial table view
- Using the Structure view to sort data, and hide and show columns
- Using the quick documentation view
- Transposing the table
- Enabling coding assistance for a column
- Modifying cell contents
- Adding a row
- Deleting rows
- Managing database transactions
- Comparing tables
- Copying table data to the clipboard or saving them in a file
- Copying a selection to the clipboard
- Changing data output format
- Saving a LOB in a file
- Updating the table view
- Viewing the query

Hiding or showing the toolbar

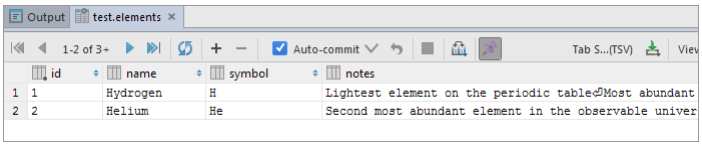
To hide or show the toolbar of the Result pane and also that of the Database Console tool window:

- Click on the title bar of the Database Console tool window and click Show Toolbar.

Pinning the Result tab

If one and the same tab is used to show your query results, and you get the result that you want to keep, you can pin the tab to the tool window. Do one of the following:

- Right-click the tab and select Pin Tab.
- Click on the toolbar.



See also, Show query results in new tab .

Switching between subsets of rows

If only a subset of the rows that satisfy the query is currently shown, to switch between the subsets, use:

- First Page
- Previous Page (`Ctrl+Alt+Up`)
- Next Page (`Ctrl+Alt+Down`)
- Last Page

See also, Making all rows visible simultaneously.

Making all rows visible simultaneously

If you want all the rows that satisfy the query to be shown simultaneously:

1. Click on the toolbar of the Database Console tool window.
2. Switch to the Database | Data Views page, specify `0` in the Result set page size field, and click OK.
3. Click or press `Ctrl+F5` to refresh the table view.

See also, Updating the table view and Result set page size .

Navigating to a specified row

To switch to a row with a specified number:

1. Do one of the following:

- Press `Ctrl+G`.
- Right-click the table and select Go To | Row from the context menu.
- Select Navigate | Row from the main menu.

2. In the dialog that opens, specify the row number and click OK.

Navigating to related records

If a row references a record in a different table or is referenced in a different table, you can switch to the corresponding table to see the related record or records.

To switch to a referenced row:

1. Do one of the following:
   - Press `Ctrl+B`.
   - Select Go To | Referenced Data from the context menu.
2. If more than one record is referenced, select the target record in the pop-up that appears.

To switch to a row that references the current one, or to see all the rows that reference the current one:

1. Do one of the following:
   - Press `Alt+F7`.
   - Select Go To | Referencing Data from the context menu.
2. Select the target in one of the following categories:
   - First Referencing Row. All the rows in the corresponding table will be shown and the first of the rows that references the current row will be selected.
   - All Referencing Rows. Only the rows that reference the current row will be shown.

The options described above can also be accessed by using one of the following:

- `F4`.
- Go To | Related Data in the context menu.
- Navigate | Related Data in the main menu.

Sorting data

You can sort table data by any of the columns by clicking the cells in the header row.

Each cell in this row has a sorting marker in the right-hand part and, initially, a cell may look something like this: . The sorting marker in this case indicates that the data is not sorted by this column.

If you click the cell once, the data is sorted by the corresponding column in the ascending order. This is indicated by the sorting marker appearance: . The number to the right of the marker (1 on the picture) is the sorting level. (You can sort by more than one column. In such cases, different columns will have different sorting levels.)

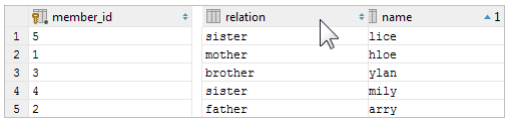
When you click the cell for the second time, the data is sorted in the descending order. Here is how the sorting marker indicates this order: .

Finally, when you click the cell for the third time, the initial state is resorted. That is, sorting by the corresponding column is canceled: .

See also, Restoring the initial table view and Using the Structure view to sort data, and hide and show columns.

Reordering columns

To reorder columns, use drag-and-drop for the corresponding cells in the header row.



	member_id	relation	name
1	5	sister	lice
2	1	mother	hloe
3	3	brother	ylan
4	4	sister	mily
5	2	father	arry

See also, Restoring the initial table view.

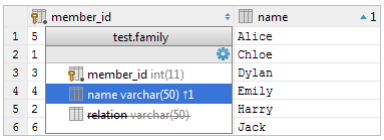
Hiding and showing columns

To hide a column, right-click the corresponding header cell and select Hide column.

To show a hidden column:

1. Do one of the following:
   - Right-click any of the cells in the header row and select Column List.
   - Press `Ctrl+F12`.

   In the list that appears, the names of hidden columns are shown struck through.



2. Select (highlight) the column name of interest and press `Space`.
3. Press `Enter` or `Escape` to close the list.

See also, Restoring the initial table view and Using the Structure view to sort data, and hide and show columns.

Restoring the initial table view

Click on the toolbar and select Reset View to restore the initial table view after reordering or hiding the columns, or sorting the data. As a result, the data, generally, becomes unsorted, the columns appear in the order they are defined in the corresponding query, and all the columns are shown.

Using the Structure view to sort data, and hide and show columns

When working with the Result pane, the table structure view is available as the corresponding popup.

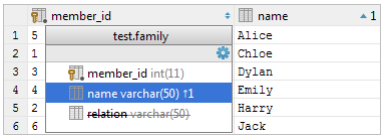
The structure view shows the list of all the columns and lets you sort the data as well as hide and show the columns.

To open the structure popup, do one of the following:

- Right-click a cell in the table header row and select Column List.
- Press `Ctrl+F12`.

In the popup, select the column of interest and do one of the following:

- To sort the data by this column in the ascending order, press `Shift+Alt+Up`.
- To sort the data in the descending order, press `Shift+Alt+Down`.
- To cancel sorting by this column, press `Ctrl+Shift+Alt+Backspace`.
- To hide the column (or show a hidden column), press `Space`. (The names of hidden columns are shown struck through.)

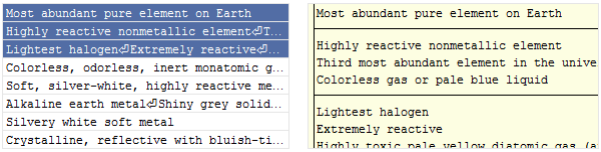


The shortcuts for sorting table data (`Shift+Alt+Up`, `Shift+Alt+Down` and `Ctrl+Shift+Alt+Backspace`) can be used in the Result pane without opening the structure view.

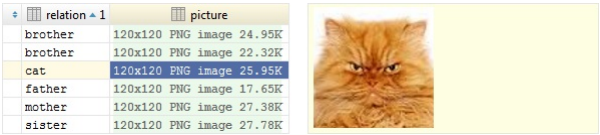
See also, Sorting data, Hiding and showing columns and Restoring the initial table view.

Using the quick documentation view

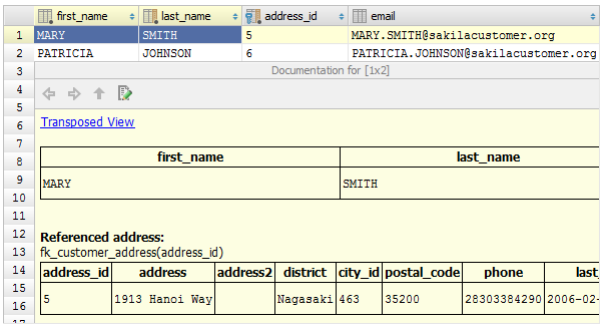
The quick documentation view provides details about the values in the selected cell or cells. For example, if a cell contains long text, normally, you can see only its beginning. The whole text is shown in the quick documentation view.



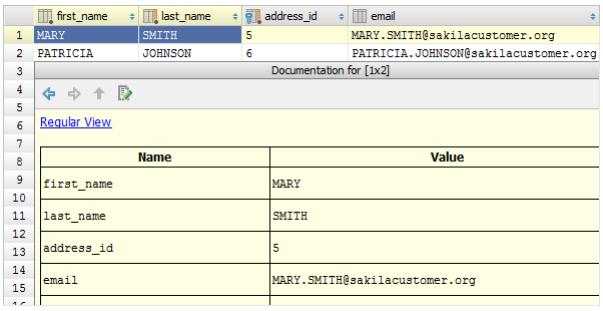
If a cell contains an image, you can see that image in the quick documentation view.



You can also see the records referenced in the current record as well as the records that reference the current one.



If necessary, you can switch to the transposed view. This is when the rows and columns are interchanged. Thus, for a row, the cells are shown one beneath the other.



To open the quick documentation view, press `Ctrl+Q` or select Quick Documentation from the View or the context menu.

To switch to the transposed view, click Transposed View. See also, Transposing the table.

To close the quick documentation view, press `Escape`.

Transposing the table

The transposed table view is available. In this view, the rows and columns are interchanged.

To turn this view on or off, click on the toolbar and select Transpose. Alternatively, use the Transpose context menu command.

Enabling coding assistance for a column

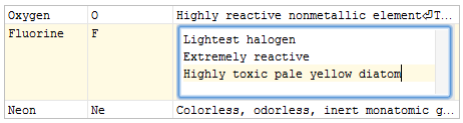
You can assign a column one of the supported languages (e.g. SQL, HTML or XML): right-click the corresponding header cell, select Edit As and select the language. As a result, you get coding assistance for the selected language in all the cells of the corresponding column.

You can also assign a language to an individual cell .

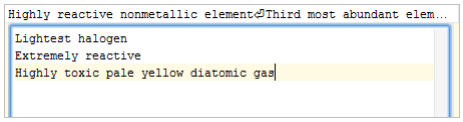
Modifying cell contents

You can modify values in the table cells and, if appropriate, upload files.

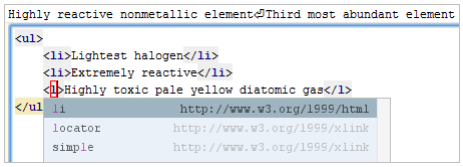
1. To start editing a value or uploading a file, do one of the following:
   - Double-click the corresponding table cell.
   - Right-click the cell and select Edit Value or Edit Maximized from the context menu.
   - Select the cell and press `F2` or `Shift+Enter`. In the latter case, the cell will be maximized.
   - Select the cell and start typing. Note that in this case the initial cell contents are deleted right away and is replaced with the typed value.
2. When in the editing mode, you can:
   - Modify the value right in the cell. To start a new line, use `Ctrl+Enter`. To save the changes, press `Enter`. To restore an initial value and quit the editing mode, press `Escape`.



   - Maximize the cell if you need more room for editing. To do that, press `Ctrl+Shift+Alt+M`, or right-click the cell and select Maximize.
     When working in a maximized cell, use `Enter` to start a new line and `Ctrl+Enter` to save the value. To restore an initial value and quit the editing mode, press `Escape`.



   - Upload a file into the field (e.g. to replace an existing file with a new one). To do that, right-click the cell and select Load File. Then select the necessary file in the dialog that opens.
   - Replace the current value with `null`. To do that, right-click the cell and select Set NULL.
   - Edit a value in the cell as a fragment in one of the supported languages (e.g. SQL, HTML or XML). To do that, right-click the cell, select Edit As and select the language. As a result, you get coding assistance for the language you have selected.



Adding a row

If + on the toolbar is enabled, you can add rows to the table.

1. To start adding a row, do one of the following:
   - Click + on the toolbar.
   - Right-click the table and select Add New Row from the context menu.
   - Press `Alt+Insert`.

   Note that the context menu Clone Row command (`Ctrl+D`) can be used as an alternative.
2. Enter the values into the cells. For instructions, see Modifying cell contents.
3. To save the new row, select Submit New Row from the context menu or press `Ctrl+Enter`.

Deleting rows

If – on the toolbar is enabled, you can delete rows. To do that:

1. Select the row or rows that you want to delete.
   Rows are selected by clicking the cells in the column where the row numbers are shown. To select more than one row, use mouse clicks in combination with the `Ctrl` key.
2. Do one of the following:
   - Click – on the toolbar.
   - Press `Ctrl+Y` or `Delete`.
3. Confirm you intention to delete the selected row or rows.

Managing database transactions

The Auto-commit check box, and the Commit and Rollback icons on the toolbar let you manage database transactions.

The Auto-commit check box is used to turn the autocommit mode for the database connection on or off.

In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back.

(Each change of a value in the table, or adding or deleting a row results in executing an SQL statement.)

If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the Commit ( ) or Rollback ( ) command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements.

In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.

Comparing tables

You can compare the current table with any other table which is open in the table editor or shown in the Database Console tool window. To do that, click on the toolbar and select the table of interest.

The comparison results are shown in the differences viewer .

Copying table data to the clipboard or saving them in a file

1. Do one of the following:
   - Click on the toolbar, and select To File or To Clipboard.
   - Right-click the table, point to Dump Data and select To File or To Clipboard.
2. If you are saving the data in a file, select the output format (the extractor), and then specify the file name and location.

If only a subset of rows is currently shown, all the rows that satisfy the corresponding query are copied to the clipboard or saved in a file anyway.

See also, Changing data output format.

Copying a selection to the clipboard

To copy selected cells or rows to the clipboard, press `Ctrl+C` or select Copy from the main or the context menu.

If the table is shown transposed (see Transposing the table), the range of cells is copied transposed (as shown) if the Allow Transposition option is on and non-transposed (as in the original table) otherwise.

To turn the Allow Transposition option on or off, right-click the table, point to Data Extractor: <current_format> and click Allow Transposition. Alternatively, click Tab S...(TSV) on the toolbar and click Allow Transposition.

Changing data output format

When you copy table data to the clipboard (the Copy and Dump Data | To Clipboard commands), unless you change the format, the data are output as Tab-Separated Values (TSV).

To change the output format or to define a new format for the output:

1. Do one of the following:
   - Click Tab S...(TSV) on the toolbar.
   - Right-click the table and point to Data Extractor: <current_format>.
2. Do one of the following:
   - To select a different output format, click the name of the format.
   - To define a new delimiter-separated values format, or to view or modify the settings for an existing one, click Configure CSV Formats. As a result, the CSV Formats Dialog will open.

Saving a LOB in a file

If a cell contains a binary large object (a.k.a. BLOB or LOB), you can save such a LOB in a file.

1. Right-click the cell that contains the LOB of interest and select Save LOB To File.
2. In the dialog that opens, specify the name and location of the destination file and click OK.

Updating the table view

To refresh the table view, do one of the following:

- Click on the toolbar.
- Right-click the table and select Reload Page from the context menu.
- Press `Ctrl+F5`.

Use this function to:

- Synchronize the data shown with the actual contents of the database.
- Apply the Result set page size setting after its change.

Viewing the query

To see the query that was used to generate the table:

- Click View Query on the toolbar.
  If necessary, you can select the query text and copy it to the clipboard (`Ctrl+C`).

To close the pane where the query is shown, press `Escape`.

Working with the Table Editor

The Table Editor provides a graphical interface for working with table data. You can sort, filter, add, edit and remove the data as well as perform other, associated tasks.

- Opening a table in the Table Editor
- Protecting a table from accidental modifications
- Switching between subsets of rows
- Making all rows visible simultaneously
- Navigating to a specified row
- Navigating to related records
- Sorting data
- Filtering data
- Using quick filtering options
- Reordering columns
- Hiding and showing columns
- Restoring the initial table view
- Using the Structure view to sort data, and hide and show columns
- Using the quick documentation view
- Transposing the table
- Enabling coding assistance for a column
- Modifying cell contents
- Adding a row
- Deleting rows
- Managing database transactions
- Comparing tables
- Copying table data to the clipboard or saving them in a file
- Copying a selection to the clipboard
- Changing data output format
- Saving a LOB in a file
- Updating the table view
- Viewing the query
- Working with the CREATE TABLE statement

Opening a table in the Table Editor

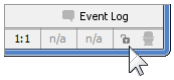
In the Database tool window , do one of the following:

- Double-click the table of interest.
- Click the table and click on the toolbar (if the toolbar is not currently hidden).
- Select the table and press `F4`.
- Right-click the table and select Table Editor from the context menu.

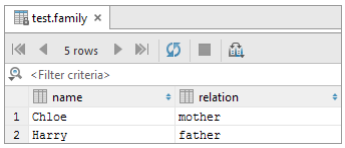
As a result, the table opens in the Table Editor on a separate editor tab.

Protecting a table from accidental modifications

To protect a table from accidental modifications in the Table Editor, you can make it read-only. To do that, click the padlock icon in the lower-right corner of PhpStorm workspace.



As a result, the icon appearance will change to , a padlock will appear on the corresponding editor tab, and you won't be able to make changes to the table.



To turn off the table's read-only status, click the padlock icon again.

Note that the tables with the read-only status in the Table Editor can still be modified when using the database console or in the Database tool window.

Switching between subsets of rows

If only a subset of all the rows is currently shown, to switch between the subsets, use:

- First Page
- Previous Page (`Ctrl+Alt+Up`)
- Next Page (`Ctrl+Alt+Down`)
- Last Page

See also, Making all rows visible simultaneously.

Making all rows visible simultaneously

If you want all the rows to be shown simultaneously:

- Click on the toolbar and select Settings

1. Click on the toolbar and select Settings.
2. Switch to the Database | Data Views page, specify `0` in the Result set page size field, and click OK.
3. Click or press `Ctrl+F5` to refresh the table view.

See also, Updating the table view and Result set page size .

Navigating to a specified row

To switch to a row with a specified number:

1. Do one of the following:
   - Press `Ctrl+G`.
   - Right-click the table and select Go To | Row from the context menu.
   - Select Navigate | Row from the main menu.
2. In the dialog that opens, specify the row number and click OK.

Navigating to related records

If a row references a record in a different table or is referenced in a different table, you can switch to the corresponding table to see the related record or records.

To switch to a referenced row:

1. Do one of the following:
   - Press `Ctrl+B`.
   - Select Go To | Referenced Data from the context menu.
2. If more than one record is referenced, select the target record in the pop-up that appears.

To switch to a row that references the current one, or to see all the rows that reference the current one:

1. Do one of the following:
   - Press `Alt+F7`.
   - Select Go To | Referencing Data from the context menu.
2. Select the target in one of the following categories:
   - First Referencing Row. All the rows in the corresponding table will be shown and the first of the rows that references the current row will be selected.
   - All Referencing Rows. Only the rows that reference the current row will be shown.

The options described above can also be accessed by using one of the following:

- `F4`.
- Go To | Related Data in the context menu.
- Navigate | Related Data in the main menu.

Sorting data

You can sort table data by any of the columns by clicking the cells in the header row.

Each cell in this row has a sorting marker in the right-hand part and, initially, a cell may look something like this: name . The sorting marker in this case indicates that the data is not sorted by this column.

If you click the cell once, the data is sorted by the corresponding column in the ascending order. This is indicated by the sorting marker appearance: name 1 . The number to the right of the marker (1 on the picture) is the sorting level. (You can sort by more than one column. In such cases, different columns will have different sorting levels.)

When you click the cell for the second time, the data is sorted in the descending order. Here is how the sorting marker indicates this order: name 1 .

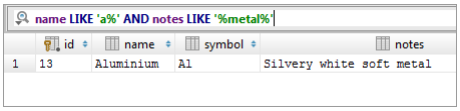
Finally, when you click the cell for the third time, the initial state is resorted. That is, sorting by the corresponding column is canceled: name .

You can turn on the Sort via ORDER BY option , to enable sorting the data by the corresponding DBMS.

See also, Restoring the initial table view and Using the Structure view to sort data, and hide and show columns.

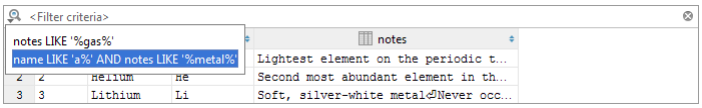
Filtering data

1. If the filter box is not currently shown, click on the toolbar and select Row Filter.
2. In the filter box, specify filtering conditions.
   The filtering conditions are specified as in a `WHERE` clause but without the word `WHERE`, e. g. `name LIKE 'a%' AND notes LIKE '%metal%'`. Within the `LIKE` expressions, the SQL wildcards can be used: the percent sign (`%`) for zero or more characters and underscore (`_`) for a single character.



   To apply the conditions currently specified in the box, press `Enter`. To cancel filtering, delete the contents of the filter box and press `Enter`.

   To reapply a memorized filter, click and select the filter in the list. See also, Filter history size .



   To cancel filtering and hide the filter box, click .

Using quick filtering options

In addition to specifying filtering conditions manually (see Filtering data), you can use quick filtering options.

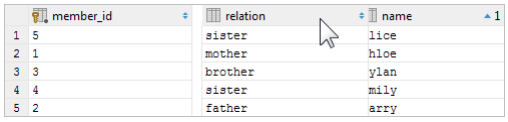
Available as context menu commands, these options are a set of filtering conditions for the current column name. The conditions themselves depend on the value in the current cell.

To use a quick filtering option:

1. Right-click a cell of interest and point to Filter by.
2. Select the necessary condition from the list.

Reordering columns

To reorder columns, use drag-and-drop for the corresponding cells in the header row.



	member_id	relation	name
1	5	sister	lice
2	1	mother	hloe
3	3	brother	ylan
4	4	sister	mily
5	2	father	arry

See also, Restoring the initial table view.

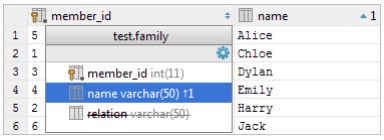
Hiding and showing columns

To hide a column, right-click the corresponding header cell and select Hide column.

To show a hidden column:

1. Do one of the following:
   - Right-click any of the cells in the header row and select Column List.
   - Press `Ctrl+F12`.

   In the list that appears, the names of hidden columns are shown struck through.



2. Select (highlight) the column name of interest and press `Space`.
3. Press `Enter` or `Escape` to close the list.

See also, Restoring the initial table view and Using the Structure view to sort data, and hide and show columns.

Restoring the initial table view

Click on the toolbar and select Reset View to restore the initial table view after reordering or hiding the columns, or sorting the data. As a result, the data, generally, becomes unsorted, the columns appear in the order they are defined in the database, and all the columns are shown.

Using the Structure view to sort data, and hide and show columns

When working with the Table Editor, the table structure view is available in the Structure tool window or as the corresponding popup.

The structure view shows the list of all the columns and lets you sort the data as well as hide and show the columns.

To open the Structure tool window, do one of the following:

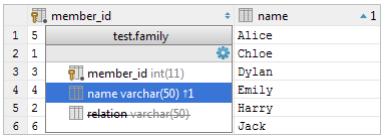
- Select View | Tool Windows | Structure in the main menu.
- Click Structure on the left-hand tool window bar .
- Press `Alt+7`.

To open the structure popup, do one of the following:

- Right-click a cell in the table header row and select Column List.
- Press `Ctrl+F12`.

In the tool window or the popup, select the column of interest and do one of the following:

- To sort the data by this column in the ascending order, press `Shift+Alt+Up`. (In the tool window, you can, alternatively, select Sort | Ascending from the context menu.)
- To sort the data in the descending order, press `Shift+Alt+Down`. (In the tool window, alternatively, Sort | Descending.)
- To cancel sorting by this column, press `Ctrl+Shift+Alt+Backspace`. (In the tool window, alternatively, Sort | Unsorted.)
- To hide the column (or show a hidden column), press `Space`. (The names of hidden columns are shown struck through. In the tool window, alternatively, the Hide Column or Show Column context menu command can be used.)

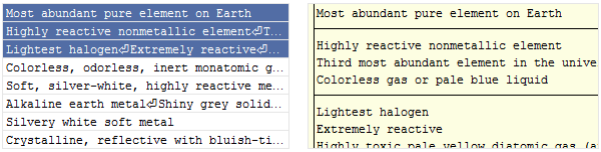


The shortcuts for sorting table data (`Shift+Alt+Up`, `Shift+Alt+Down` and `Ctrl+Shift+Alt+Backspace`) can be used in the Table Editor without opening the structure view.

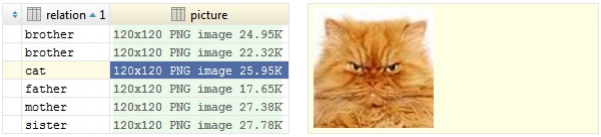
See also, Sorting data, Hiding and showing columns and Restoring the initial table view.

Using the quick documentation view

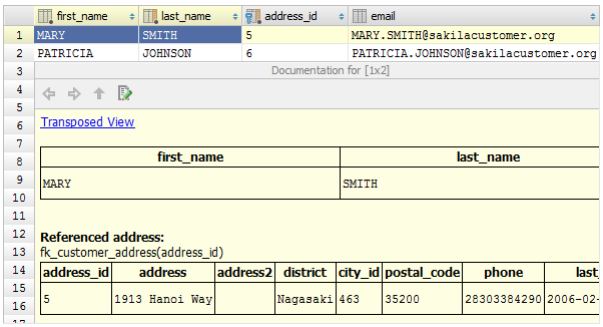
The quick documentation view provides details about the values in the selected cell or cells. For example, if a cell contains long text, normally, you can see only its beginning. The whole text is shown in the quick documentation view.



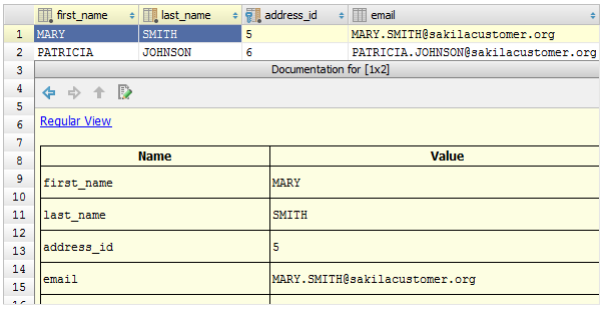
If a cell contains an image, you can see that image in the quick documentation view.



You can also see the records referenced in the current record as well as the records that reference the current one.



If necessary, you can switch to the transposed view. This is when the rows and columns are interchanged. Thus, for a row, the cells are shown one beneath the other.



To open the quick documentation view, press `Ctrl+Q` or select Quick Documentation from the View or the context menu.

To switch to the transposed view, click Transposed View. See also, Transposing the table.

To close the quick documentation view, press `Escape`.

Transposing the table

The transposed table view is available. In this view, the rows and columns are interchanged.

To turn this view on or off, click on the toolbar and select Transpose. Alternatively, use the Transpose context menu command.

Enabling coding assistance for a column

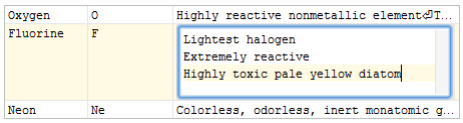
You can assign a column one of the supported languages (e.g. SQL, HTML or XML): right-click the corresponding header cell, select Edit As and select the language. As a result, you get coding assistance for the selected language in all the cells of the corresponding column.

You can also assign a language to an individual cell .

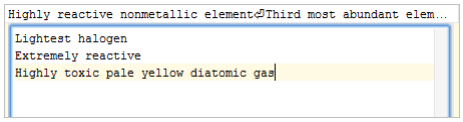
Modifying cell contents

You can modify values in the table cells and, if appropriate, upload files.

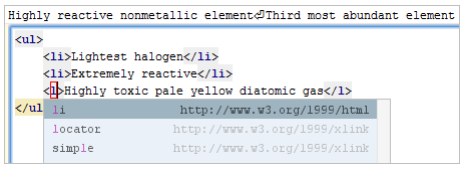
1. To start editing a value or uploading a file, do one of the following:
   - Double-click the corresponding table cell.
   - Right-click the cell and select Edit Value or Edit Maximized from the context menu.
   - Select the cell and press `F2` or `Shift+Enter`. In the latter case, the cell will be maximized.
   - Select the cell and start typing. Note that in this case the initial cell contents are deleted right away and is replaced with the typed value.
2. When in the editing mode, you can:
   - Modify the value right in the cell. To start a new line, use `Ctrl+Enter`. To save the changes, press `Enter`. To restore an initial value and quit the editing mode, press `Escape`.



- Maximize the cell if you need more room for editing. To do that, press `Ctrl+Shift+Alt+M`, or right-click the cell and select Maximize.
  When working in a maximized cell, use `Enter` to start a new line and `Ctrl+Enter` to save the value. To restore an initial value and quit the editing mode, press `Escape`.



- Upload a file into the field (e.g. to replace an existing file with a new one). To do that, right-click the cell and select Load File. Then select the necessary file in the dialog that opens.
- Replace the current value with `null`. To do that, right-click the cell and select Set NULL.
- Edit a value in the cell as a fragment in one of the supported languages (e.g. SQL, HTML or XML). To do that, right-click the cell, select Edit As and select the language. As a result, you get coding assistance for the language you have selected.



Adding a row

If + on the toolbar is enabled, you can add rows to the table.

1. To start adding a row, do one of the following:
   - Click + on the toolbar.
   - Right-click the table and select Add New Row from the context menu.
   - Press `Alt+Insert`.

   Note that the context menu Clone Row command (`Ctrl+D`) can be used as an alternative.
2. Enter the values into the cells. For instructions, see Modifying cell contents.
3. To save the new row, select Submit New Row from the context menu or press `Ctrl+Enter`.

Deleting rows

If – on the toolbar is enabled, you can delete rows. To do that:

1. Select the row or rows that you want to delete.
   Rows are selected by clicking the cells in the column where the row numbers are shown. To select more than one row, use mouse clicks in combination with the `Ctrl` key.
2. Do one of the following:
   - Click – on the toolbar.
   - Press `Ctrl+Y` or `Delete`.
3. Confirm you intention to delete the selected row or rows.

Managing database transactions

The Auto-commit check box, and the Commit and Rollback icons on the toolbar let you manage database transactions.

The Auto-commit check box is used to turn the autocommit mode for the database connection on or off.

In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back.

(Each change of a value in the table, or adding or deleting a row results in executing an SQL statement.)

If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the Commit ( ) or Rollback ( ) command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements.

In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.

Comparing tables

You can compare the current table with any other table which is open in the table editor or shown in the Database Console tool window. To do that, click on the toolbar and select the table of interest.

The comparison results are shown in the differences viewer .

Copying table data to the clipboard or saving them in a file

1. Do one of the following:
   - Click on the toolbar, and select To File or To Clipboard.
   - Right-click the table, point to Dump Data and select To File or To Clipboard.
2. If you are saving the data in a file, select the output format (the extractor), and then specify the file name and location.

If only a subset of rows is currently shown, all the rows are copied to the clipboard or saved in a file anyway.

See also, Changing data output format.

Copying a selection to the clipboard

To copy selected cells or rows to the clipboard, press `Ctrl+C` or select Copy from the main or the context menu.

If the table is shown transposed (see Transposing the table), the range of cells is copied transposed (as shown) if the Allow Transposition option is on and non-transposed (as in the original table) otherwise.

To turn the Allow Transposition option on or off, right-click the table, point to Data Extractor: <current_format> and click Allow Transposition. Alternatively, click Tab S...(TSV) on the toolbar and click Allow Transposition.

Changing data output format

When you copy table data to the clipboard (the Copy and Dump Data | To Clipboard commands), unless you change the format, the data are output as Tab-Separated Values (TSV).

To change the output format or to define a new format for the output:

1. Do one of the following:
   - Click Tab S...(TSV) on the toolbar.
   - Right-click the table and point to Data Extractor: <current_format>.
2. Do one of the following:
   - To select a different output format, click the name of the format.
   - To define a new delimiter-separated values format, or to view or modify the settings for an existing one, click Configure CSV Formats. As a result, the CSV Formats Dialog will open.

Saving a LOB in a file

If a cell contains a binary large object (a.k.a. BLOB or LOB), you can save such a LOB in a file.

1. Right-click the cell that contains the LOB of interest and select Save LOB To File.
2. In the dialog that opens, specify the name and location of the destination file and click OK.

Updating the table view

To refresh the table view, do one of the following:

- Click on the toolbar.
- Right-click the table and select Reload Page from the context menu.
- Press `Ctrl+F5`.

Use this function to:

- Synchronize the data shown with the actual contents of the database.
- Apply the Result set page size setting after its change.

Viewing the query

To see the query that was used to generate the table:

- Click View Query on the toolbar.
  If necessary, you can select the query text and copy it to the clipboard (`Ctrl+C`).

To close the pane where the query is shown, press `Escape`.

Working with the CREATE TABLE statement

Click the DDL tab (in the lower-left part of the Table Editor) to see the `CREATE TABLE` statement used to create the table.

If necessary, you can edit this statement and then run it (on the toolbar or `Ctrl+Shift+F10`).

To regenerate the `CREATE TABLE` statement for the current state of the table in the database, use or `Ctrl+F5`.

Running SQL Script Files

You can run an SQL file as a whole. You can also execute individual statements contained in an SQL file.

- Running an SQL file
- Executing individual statements

Running an SQL file

When running an SQL script file as a whole:

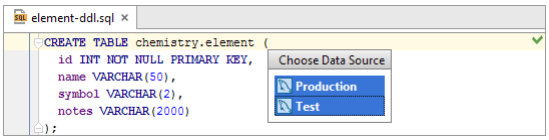
- You don't need to open the file in the editor. You can select the necessary file in the Project tool window.
- You can run the file for more than one data source at once.

On the other hand:

- The statements with parameters won't run.
- Retrieved data for the `SELECT` statements won't be shown.

To run an SQL file:

1. Select the necessary SQL file in the Project tool window, or open the file in the editor.
2. Do one of the following:
   - Select Run "<file_name>" from the context menu.
   - Press `Ctrl+Shift+F10`.
3. In the Choose Data Source pop-up, click the data source to which the script should be applied.
   If you want to run the script for more than one data source, select the data sources of interest in the pop-up and press `Enter`.



Executing individual statements

When running individual statements contained in an SQL file:

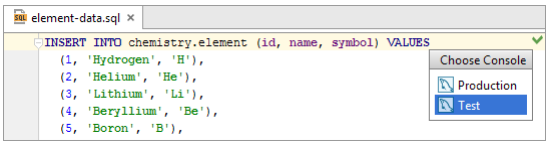
- The statements can contain parameters. Prior to running such statements PhpStorm will ask you to specify the parameter values.

On the other hand:

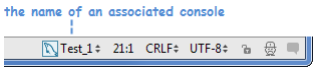
- The statements are run for only one data source at a time.

To run a statement or statements:

1. Open the SQL file of interest in the editor.
2. Place the cursor within the statement you want to execute.
   If you want to run more than one statement, select (highlight) the necessary statements.
3. Do one of the following:
   - Press `Ctrl+Enter` or select Execute from the context menu.
   - Press `Alt+Enter` or click , and select Run query in console.
4. Select the database console to be used.



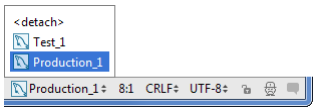
The statement or statements are executed using the selected console. The corresponding console is associated with the file. The name of the associated console is shown on the Status bar.



After that, the Execute and the Run query in console commands start to work differently:

- When using the Execute command, the associated console is used.
- When using the Run query in console command, PhpStorm asks you to select the console (as before).

You can use the menu on the Status bar to associate the file with a different console.



<detach> in this menu removes the association between the file and the console.

For the Execute command options, see Execute in Console.

Running Injected SQL Statements

You can inject an SQL statement into a string literal (see Using Language Injections) and then run that statement:

1. In the editor, place the cursor within the corresponding string literal.
2. Do one of the following:
   - Press `Ctrl+Enter`.
   - Press `Alt+Enter` and select Run query in console.
   - Click and select Run query in console.
3. If asked, select the database console to be used.
4. If the statement contains parameters, specify the parameter values.

Configuring sprintf Format References

You can have PhpStorm recognize sprintf format references in injected SQL strings:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Tools node, and then click User Parameters under Database.
2. Select the Enable in console and SQL files and Enable in string literals with SQL injection check boxes.
3. In the Parameter Patterns list:
   1. Click the `%w+` pattern. PhpStorm expands a hidden area with the configuration settings for the pattern.
   2. Select the In scripts and In literals check boxes.
   3. Click the <Language> link and select PHP in the drop-down list.

Docker

- Prerequisites
- Overview of Docker support
- Working with Docker: Process overview
- PhpStorm and Docker integration configuration
- Launching default Docker machine and getting necessary parameters
- Configuring PhpStorm to work with Docker
- Creating a Docker Deployment run/debug configuration
- Working with Docker in PhpStorm
- Running Docker from PhpStorm
- Managing Docker containers and other Docker-related actions in PhpStorm
- Debugging PHP web application running in the Docker container

Prerequisites

Make sure that the following prerequisites are met:

- Docker is installed, as described on the page Docker Docs . You can install Docker on the various platforms:
  - Windows
  - Mac OS X
  - Linux (Ubuntu, other distributions-related instructions are available as well);
  - Cloud platforms (e.g. here are tutorials on installing Docker on Amazon EC2 , Google Cloud Platform , and Microsoft Azure .)
- Plugin Docker integration is installed.

  The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Overview of Docker support

The Docker integration plugin adds the following to PhpStorm:

- Docker configurations. These are named sets of settings for accessing the Docker Remote API.
- Docker Deployment run/debug configurations . They let you download and build Docker images, and create and start Docker containers. Application debugging is supported only for PHP.
- Docker tool window that lets you manage your Docker images and containers.
- Docker Registry configurations that represent your Docker image repository user accounts.

Working with Docker: Process overview

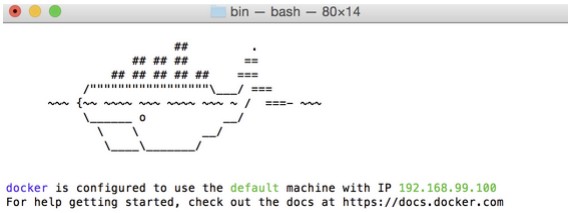
1. Install Docker and start the Docker daemon (on Windows and OS X - along with the Docker VM in which it runs). For corresponding instructions, see Get Started with Docker .
2. Create a Docker configuration. You can do that:
   - Separately, in the Settings / Preferences dialog: `Ctrl+Alt+S` + | Docker.
   - When creating a Docker Deployment run configuration: Run | Edit Configurations | + | Docker Deployment, etc.
4. Create a Docker Deployment run configuration .
5. Execute the run configuration.
6. Use the Docker Tool Window to manage your Docker images, containers and deployed applications.

PhpStorm and Docker integration configuration

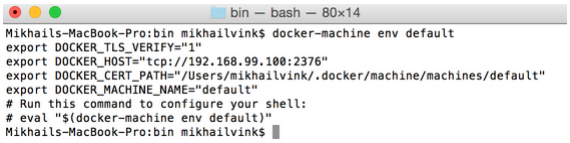
At this point you have Docker and Docker Integration plugin for PhpStorm installed, so we can start with integration configuration.

Launching default Docker machine and getting necessary parameters

1. Open the Docker Quickstart Terminal application. Docker will launch a default VM providing output in the console:

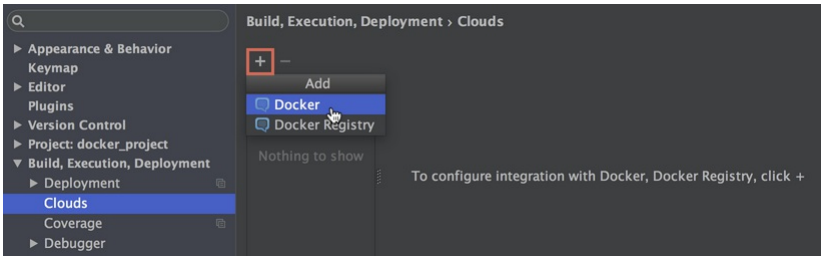


2. In the Docker console, run the console command `docker-machine env default` . Thus you will get all the necessary parameters - you'll need them later for PhpStorm configuration (such as `DOCKER_HOST` and `DOCKER_CERT_PATH`):

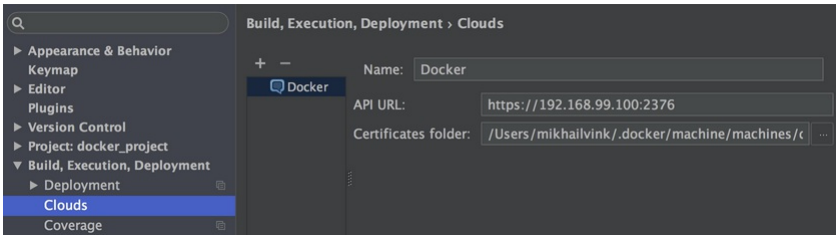


Configuring PhpStorm to work with Docker

1. In the Settings / Preferences dialog, open the Docker page under Build, Execution, Deployment, and click + to create a Docker configuration:



2. In the Docker - Docker page, provide the following information:
   - Name. Here it is Docker.
   - API URL. Here it is https://192.168.99.100:2376. This value is taken from the console output (in the second step of the previous procedure .)
     Pay attention to the protocol change: `tcp` (in the second step of the previous procedure) has changed to `https`.
   - Certificates folder. Here it is `/Users/<user name>/.docker/machine/machines/`



> **Note** On Linux/Unix, you might need to provide a socket path in the API URL field instead of the URL (usually it's unix:///var/run/docker.sock by default)

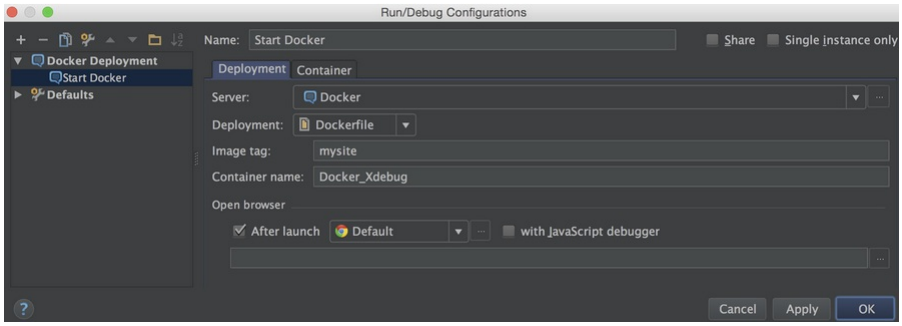
3. Apply the changes and close the Settings / Preferences dialog.

Creating a Docker Deployment run/debug configuration

Having set up Docker, create the run/debug configuration .

First, in the Deployment tab, specify the following:

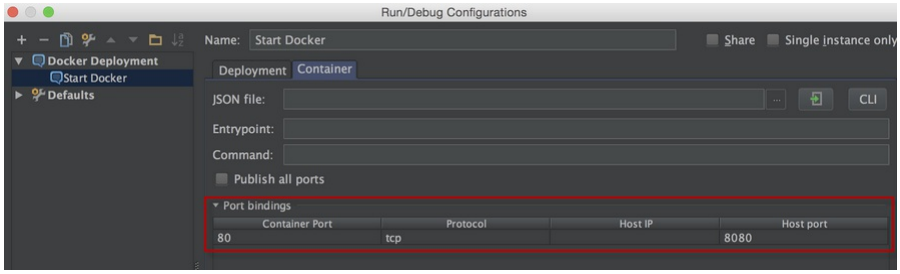
- Name. Here it is Start Docker.
- Server. Here Docker server is selected from the drop-down list.
- Deployment method. Here Dockerfile is selected.
- Image tag. Here it is mysite.
- Container name. Here it is Docker_Xdebug.



Next, in the Container tab, specify the necessary parameters.

At this point we are interested in exposing 80 port of the container to be available from our local machine, so we should configure a port binding for that:

`Container port: 80, Protocol: tcp, Host IP: empty, Host port: 8080`



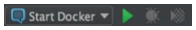
Apply changes and close the Docker Deployment run/debug configuration dialog.

> **Note** It's important to note that that there are many ways to create/configure the Docker containers and VM images, and we'd recommend to always refer to the Docker documentation on the official web site .
>
> For the purposes of this tutorial we're using Dockerfile and Apache configuration file, we've also created an `index.php` file with `phpinfo();` to be deployed to the Docker container.

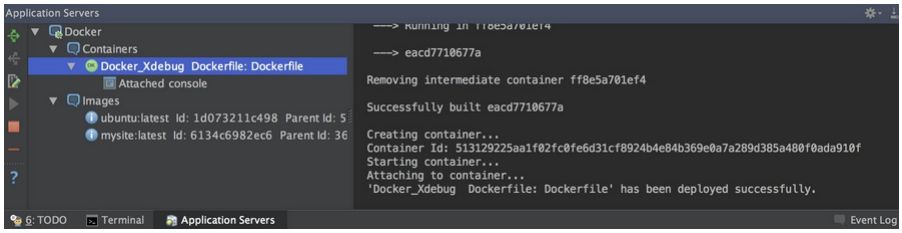
Working with Docker in PhpStorm

Running Docker from PhpStorm

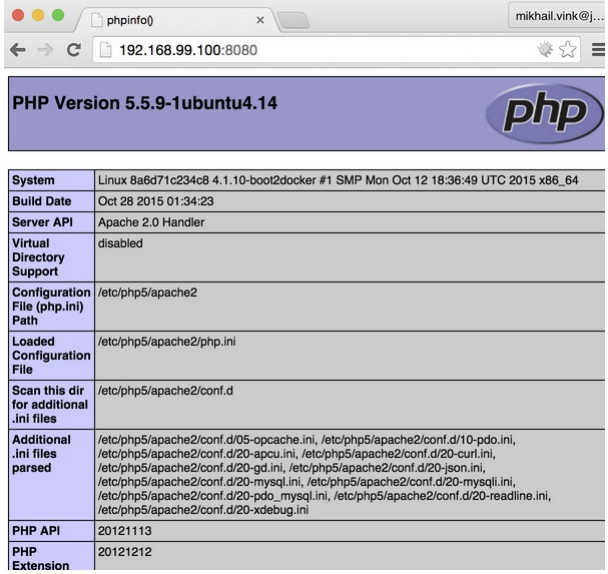
As all the tools are installed, and the integration is configured, the recently created Start Docker Run/Debug Configuration can be launched:



The Application Servers Tool Window opens, updating you on the provisioning status and the current state of all your Docker containers:

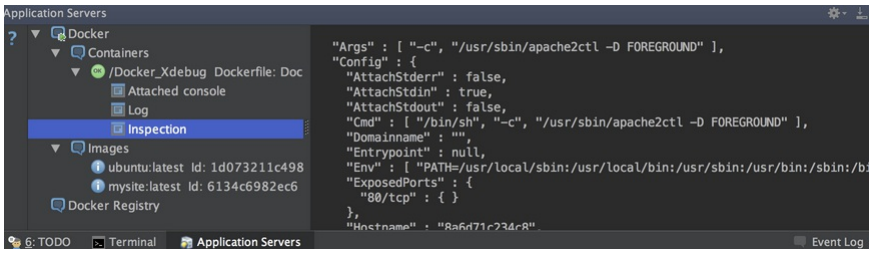


As soon as the process is completed and our Docker_Xdebug container status turns green, one can check how it works in the browser. You should be able to open it by the URL similar to `http://192.168.99.100:8080/` (`192.168.99.100` is an IP address of the default Docker machine, we've looked into where to get this parameter before). If you can't see the `index.php` execution results in the browser (that contains `phpinfo();` in our case), make sure that you have specified the correct IP address and port bindings on the previous steps. In our example, everything is running fine on the port we've expected the app to be:

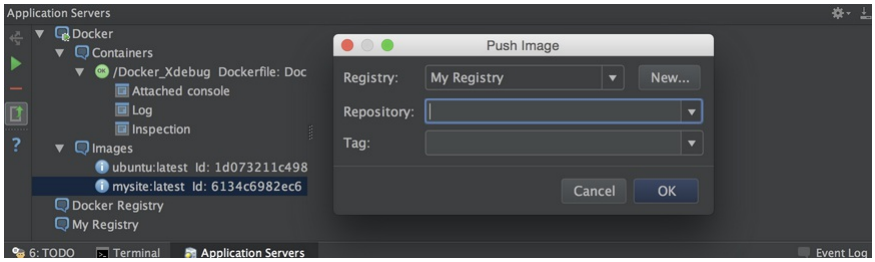


Managing Docker containers and other Docker-related actions in PhpStorm

From the Application Servers Tool Window , it's easy to inspect containers and view running processes. PhpStorm also enables you to search through the logs, start and stop containers, and perform basic container management like creating and deleting containers. Each deployment in Docker is assigned a unique container ID - these are initially temporary containers, although they can be committed and saved for further distribution. On the Docker Hub registry, there are many such images available for you to try.



Images in Docker are read-only - once committed, any changes to a container's state will become part of a new image. When you have a stable build on one instance of Docker (on your development machine, staging server, or a cloud), reproducing the exact same build is as simple as (1) committing the Docker container, (2) pushing it to a registry (public or private), then (3) pulling the same image to another instance of Docker, running - wherever.



Debugging PHP web application running in the Docker container

Assuming that you already run the Docker container now (and everything worked well on the previous steps), you should now able to open your PHP web application in the browser by `http://host:port URL (http://192.168.99.100:8080/` in our case). The major difficulty in getting Xdebug (or Zend Debugger) working with PhpStorm and Docker integration is the correct configuration of the Docker container.

In our case we're using a Dockerfile (we've already shown this config earlier and provided links to download it) to configure the container, including Xdebug-specific parameters, such as:

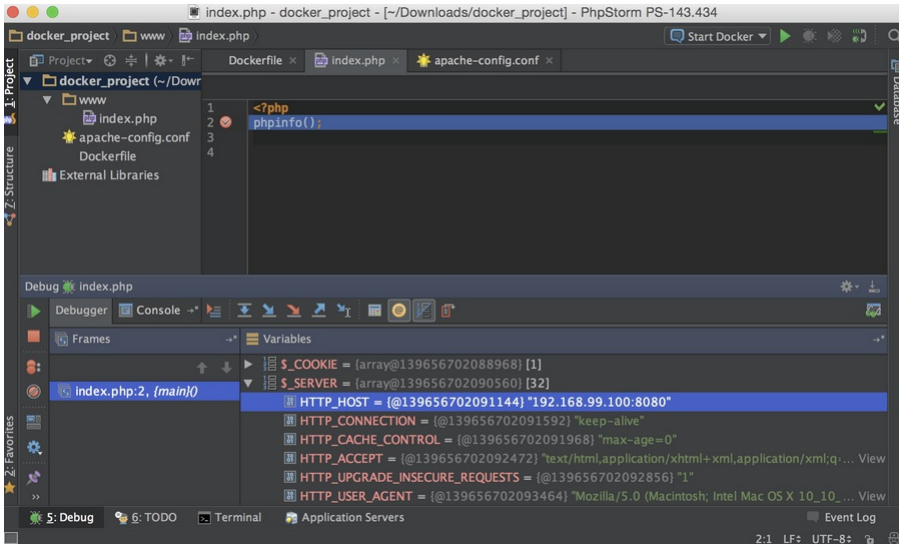
```
#Set up debugger
                    RUN echo "zend_extension=/usr/lib/php5/20131226/xdebug.so" >> /etc/php5/apache2/php.ini
                    RUN echo "xdebug.remote_enable=1" >> /etc/php5/apache2/php.ini
                    #Please provide your host (local machine IP) instead of 192.168.2.117
                    RUN echo "xdebug.remote_host=192.168.2.117" >> /etc/php5/apache2/php.ini
```

In the example above we're modifying `/etc/php5/apache2/php.ini` providing a path to Xdebug extension, and some other Xdebug parameters (remote_enable and remote_host). Please note that `xdebug.remote_host` value should be replaced with your local machine IP address (where PhpStorm is running, in our case it is `192.168.2.117`).

Configuration for Zend Debugger is similar, please see a full tutorial on installing Xdebug and installing Zend Debugger (there's more information on required parameters and options).

Don't forget to re-run Start Docker Run/Debug Configuration so that all the changes are applied.

As soon as all the configs are in place, the debugging process can be triggered following this tutorial from step 2 (start Listening for PHP Debug Connections, set a breakpoint in the source code, start a debug session in the browser, reload the current page, debug).



IntelliLang

This a third-party documentation.

The IntelliLang plugin offers a number of features related to the use of custom languages in PhpStorm.

Prerequisites

Before you start working with language injections, make sure that the IntelliLang plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

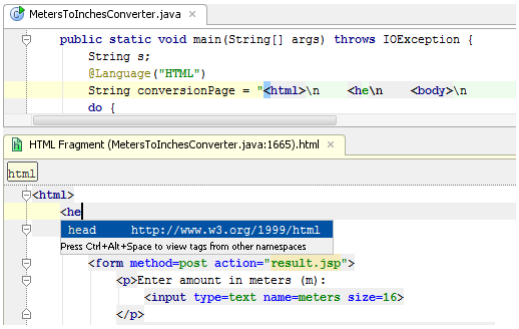
In this part:

- Overview
- General Usage
- Code Inspections
- Quick Edit Language
- Usage Examples
- IntelliLang Configuration

Overview

IntelliLang is a combination of three basic kinds of functionality that are meant to support the developer in dealing with certain tasks that relate to (custom) languages in PhpStorm:

- Language Injection: Editing the code that is embedded into String literals and XML fragments natively.
  You can open and modify an injected language code fragment in the editor . In this way you get all the necessary coding assistance, as if you were working with the source code in the corresponding language.



  To open an injected language code fragment in the editor, use the Edit <Language> Fragment intention action .
- Pattern Validation: Provides assistance in making sure that Strings being passed to and from methods match a particular regular expression
- Regular Expression Support: A custom language implementation for regular expressions

Language injection

This makes use of the new possibilities of PhpStorm to treat String literals, XML text and attributes as fragments of an arbitrary language (called *Language Injection*). The plugin makes this newly introduced API readily available to everybody for their daily use through two very simple means: Either by using some Java annotations to mark String fields, local variables, method parameters and methods returning Strings as containing a certain language, or by just using a simple UI configuration. There is a set of annotations provided by the plugin, but the actual annotations are freely configurable to avoid any unwanted dependencies.

This enables the developer to get the benefit of a wide range of edit-time features, such as syntax error highlighting, completion, inspections and a lot more while editing fragments of e.g. JavaScript inside regular Java code or in XML files of custom schemas that PhpStorm usually doesn't know about.

Pattern Validation

Additionally, the plugin allows you to annotate Java elements of type String to have them checked for compliance with certain Regular Expressions. This can be useful for very simple *languages* where the developer needs to make sure that an expression conforms to a certain syntax, e.g. that a String is a legal Java identifier or a valid printf-like pattern used by `java.util.Formatter` .

This can both be validated on-the-fly while editing the code as well as during runtime (method parameters and return value only, like the @NotNull instrumentation of PhpStorm core) by instrumenting the compiled classes with assertions that match the value against the supplied pattern.

Regular Expression Support

This part of the plugin implements language-support for `java.util.regex.Pattern` and has been mainly created to support the IntelliLang plugin by adding support for the micro-language that is probably one of the most often used one inside Strings. It features complete support for the syntax of the SDK's regular expression implementation and adds some further features, such as

- Completion and validation for character property names (e.g. `\p{javaJavaIdentifierStart}`) which nobody can usually remember anyway
- Validation and navigation for the use of back-references (e.g. `\1`), e.g ctrl-b navigates to the capturing group the backref refers to.
- Intention Actions to simplify usages of repeated character occurrences, e.g. `a{0,1}` is offered to be converted to `a?`
- "Surround With" capturing/non-capturing group
- and more

General Usage

The plugin's usage is simple and straightforward, yet very flexible. Either add the provided set of annotations to your project and start using them, configure the plugin to use a custom set of annotations or simply use the UI configuration to make PhpStorm learn that e.g. the String argument of `Pattern.compile()` should be treated as a regular expression.

Using the annotations

IntelliLang makes use of three base-annotations: `@Language`, `@Pattern` and `@Subst`

- `@Language` is responsible for the Language injection feature
- `@Pattern` is used to validate Strings against a certain regular expression pattern
- `@Subst` is used to substitute non-compile time constant expressions with a fixed value. This allows you to validate patterns and build the prefix/suffix of an injected language (see below) even for non-constant expressions that are known to contain certain kinds of values during runtime.

The annotations supplied with IntelliLang are located in the file `annotations.jar` which can be found in `/plugins/IntelliLang/lib`.

@Language

The `@Language` annotation can be used to annotate String fields, local variables, method parameters and methods returning Strings. This will cause String-literals that are assigned to a field/variable, passed as a parameter or are used as a method's return value to be interpreted as the specified language.

Additionally, there's the Language Mismatch inspection which checks for clashes between the *expected language* and the *actual language* when a field/variable is assigned or a value returned from a method.

The plugin supports *direct* and *indirect* annotations, i.e. you can either directly use the annotation like this:

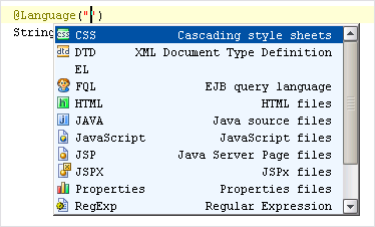
```
@Language("JavaScript")
                    String code = "var x = 1 + 2";
```

or annotate another annotation class like this:

```
@Language("XPath")
                    public @interface XPath { }
```

which can then simply be used to annotate elements as `@XPath`.

It's very easy to obtain the language-id that must be supplied as the annotation's value attribute: IntelliLang provides the list of available language via the regular code-completion action. Just select the appropriate language from the ctrl-space pop-up window:



@Pattern

The `@Pattern` annotation is responsible for marking Strings that have to comply with a certain regular expression and can be used in just the same way as the `@Language` annotation. That means, it's possible to create derived annotations, such as an `@Number` annotation that requires a String to consist of one or more digits:

```
@Pattern("\\d+")
                    public @interface Number { }
```

In fact, the predefined annotations already contain two of such derived annotations: The first one, `@PrintFormat`, matches the printf-like pattern used by `java.util.Formatter` and another one, `@Identifier`, describes a valid Java identifier. These are ready to use without having to specify any additional pattern.

@Subst

The `@Subst` annotation is used to substitute references that are not compile-time constant which enables the plugin to do Pattern Validation based on the assumption that the substituted value is compatible with the values that are expected during runtime. The plugin will complain if the value does not match the expected pattern.

It also helps to build a valid context of prefix/suffix (see next section) for the Language Injection feature. Consider this example:

```
@Subst("Tahoma")
                    final String font = new
                    JLabel().getFont().getName();

                    @Language("HTML")
                    String message = "<html><span
                    style='font: " + font + ";
                    font-size:smaller'>" + ... + "</span></html>";
```

Without substituting the value of the variable `font` with the value *Tahoma* (actually it could just be a single character here), the injected fragment would be syntactically

incorrect, causing the error *a term expected* to be displayed after the " `font:` " instruction.

Supplying context: prefix and suffix

When annotating an element, it's possible to supply a prefix and a suffix that should be prepended/appended when the language fragment is being parsed. This can be used to supply context information, i.e. if the prefix for a JavaScript injection is "var someContextVariable;", PhpStorm will know that the variable *someContextVariable* is declared and will not warn about it being undeclared when it's used.

Apart from the manual ability to supply a prefix and suffix, IntelliLang dynamically determines those values from the context a String literal is being used in:

```
@Language(value =JavaScript, prefix = "function doSomethingElse(a){
                    }")
                    String code = "function doSomething()
                    {\n" +
                    "  var x = 1;\n" +
                    "  doSomethingElse(x);\n" +
                    "}";
```

In this example, the JavaScript language will be injected into each of the three String literals, and each one's prefix and suffix will be calculated from the preceding and following expressions so that the resulting text that's being parsed is valid JavaScript syntax:

- no "missing '}'" error will be displayed: The closing brace is part of the first literal's suffix.
- the variable `x` that is used in `doSomethingElse(a);` will be declared: Its declaration is part of the second literal's prefix
- the function `doSomethingElse()` will be known as well: It's defined in the statically supplied prefix

Warning! There are some issues with the PhpStorm Language Injection API that impose certain restrictions on the prefix/suffix of an injected language fragment. For instance, it's not allowed for a token of the language to span across the prefix/suffix of an element. This could e.g. happen if the prefix ends with a whitespace character and the fragment starts with whitespace. The plugin deals with this special situation in the way that it trims the prefix/suffix and inserts exactly one space character as a separator. However, this doesn't work if a space character is no token separator, which e.g. applies to JavaScript string literals. Such cases cannot be automatically dealt with and PhpStorm core will produce an assertion.

Note Even though the dynamic prefix/suffix calculation provides a proper context for the language fragment, some things may not work as expected. Most notably this are the refactoring (rename) and navigation functions. See also the known issues below.

Code Inspections

IntelliLang contains a set of inspections that validate the correct use of the supplied annotations (or custom configured ones). These inspections can be configured through the regular Settings | Errors configuration dialog.

Language Injection

The inspections in this category apply to the language injection feature related to the `@Language` annotation.

Unknown Language ID

This inspection provides validation for the use of non-existing Language-IDs. It flags usages of incorrect values of the `@Language` value attribute, such as `@Language("NonExistingID")`.

Language Mismatch

Validation for using references to elements that are annotated as containing different languages or are not annotated at all. The inspection offers a Quick-Fix to annotate such elements with the right annotation for the expected language.

Injection not applicable

This inspection checks whether an @Language or any derived annotation is used for anything other than elements of type `String` or `String[]`.

Pattern Validation

This category contains inspections about validating the use of the @Pattern or its derived annotations.

Validate Annotated Patterns

This inspection validates that expressions (String literals, as well as other compile-time constant or substituted expressions) match the pattern required by the `@Pattern` annotation. The inspection has an option to ignore non-constant expressions that contain non-substituted references and offers a Quick-Fix to add a substitution where applicable.

Pattern Annotation not applicable

Checks whether a pattern-validation annotation (`@Pattern` or derived ones) is valid to be applied to the annotated element. Only elements of type `String` may be annotated.

Non-annotated Method overrides @Pattern Method

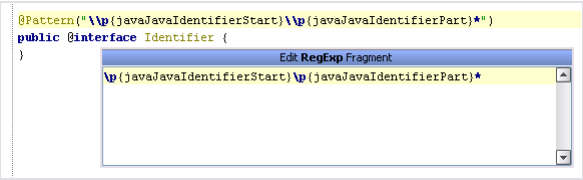
This inspection checks whether a method without any `@Pattern` or derived annotation overrides an annotated method from its base classes. This is not necessary for the error-highlighting inside the editor, however the runtime-check instrumentation doesn't pick up annotations from base-class methods.

A Quick-Fix is provided to add an annotation that matches the one from the base-class method. This ensures that the runtime-check instrumentation works correctly.

Quick Edit Language

This is an experimental feature that can be extremely helpful, especially for editing Regular Expression patterns inside String literals due to the *double escaping* requirement. For example, in a regular expression, a literal backslash character has to be written as a double backslash and each of them has to be escaped with another backslash when written inside a String literal.

The Quick Edit function that appears as an Intention Action for any injected language fragment displays a pop-up dialog that allows you to edit the string's value without the double escaping requirement. The dialog also shows the particular prefix/suffix of the fragment in a non-editable area.



The pop-up window can be dismissed by pressing Escape or clicking somewhere outside the pop-up window. Any changes made in the pop-up window are committed by pressing Ctrl-Enter.

Please note that the formatting of the non-editable prefix/suffix may differ from the actual value due to some problems. However, even though the formatting/alignment of the editable text may differ from the expected text, making changes to the text still works as expected.

Warning! Accepting Inspection Quick-Fixes inside the QuickEdit pop-up window may cause strange things to happen when the Quick-Fix attempts to open an Editor for the element being edited. The *Create Method* Quick-Fix of the JavaScript language is an example for that.

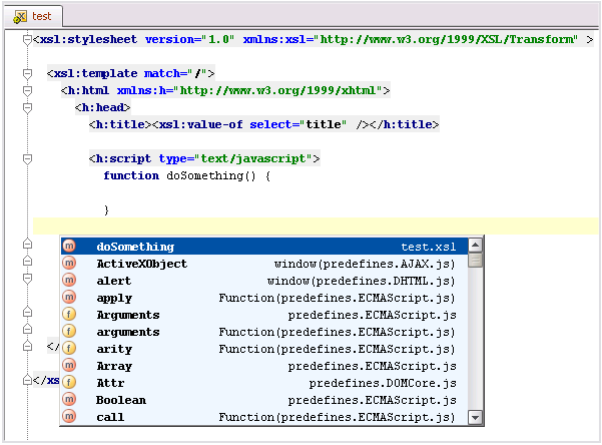
Usage Examples

Among the rather simple use-cases, like having detailed syntax checks for all kinds of "micro-languages" that are used e.g. in `Pattern.compile()`, `XPath.compile()` and so on, here are some less obvious, yet very useful examples how IntelliLang can leverage PhpStorm's support for a much better coding assistance.

The new scripting support in the upcoming Java 1.6 is another case where it will be important to get as much as possible edit-time assistance when script-code is constructed from Java code.

Extended JavaScript support

When dealing with JavaScript that's not directly embedded inside an HTML page, PhpStorm usually just treats it as plain text. Consider the following example that creates an HTML page from an XSLT script. Without the JavaScript language being injected into the *script* tag with the XHTML namespace as shown in the screenshot below, this would be treated as plain text, with no further code assistance.



Support for JSP custom tags

With IntelliLang it's also possible to make the content and attributes of custom JSP tags being treated as another language. This can be useful e.g. for server-side scripting using JavaScript or any other Language implementation available for PhpStorm.

One thing that's important to know is that the taglib's URI which supplies a custom tag should be used as the namespace URI of the XML tag to inject a language into. The namespace-textfields contain a list of all known taglib URIs in the project.

Warning! Unfortunately, at the moment the support for refactoring and navigation inside JSP custom tags seems to be broken and attempting to use code completion may result in exceptions thrown in PhpStorm core. See also the known issues below.

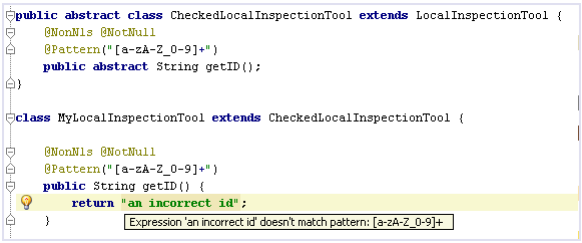
Pattern validation

Here's an obvious example right from PhpStorm's OpenAPI:

```
/** com.intellij.codeInspection.LocalInspectionTool

* @return descriptive name to be used in
suppress comments and annotations,
*         must consist of [a-zA-Z_0-9]+
*/
@NonNls @NotNull public String getID() {
return getShortName();
}
```

The contract of the method `getID()` is that it should only return strings that match the pattern "[a-zA-Z_0-9]+". The short note in the JavaDoc can be easily overlooked though because the contract isn't specified in an automatically verifiable way.

However, if this method were annotated as `@Pattern("[a-zA-Z_0-9]+")`, any attempt to return a string that doesn't match that pattern would be flagged in the editor:



Pattern completion

If a regular expression pattern represents an enumeration of different literal values, the plugin offers completion for those values:

```
void test(@Pattern("abc|xyz|123|456") String s) {...}

void doTest() {
    test("")
}
```
123
456
abc
xyz

Editing regular expressions

Here are some examples of the enhanced coding support for regular expression patterns:

Backref validation

```
@Language("RegExp")
String s =  "(abc)\\1\\2";
```
Unresolved backreference

Surround with

```
@Language("RegExp")
String s = "ab
```
Surround With

1. Capturing Group (pattern)
2. Non-Capturing Group (?:pattern)

Character category validation

```
@Language("RegExp")
String s = "\\p{xxx}";
```
Unknown character category

Character category completion

```
@Language("RegExp")
String s = "\\p{}";
```

all	ALL
Alnum	Alphanumeric characters
Alpha	Alphabetic characters
ASCII	ASCII

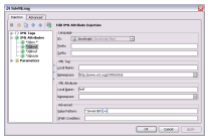
IntelliLang Configuration

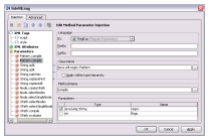
The configuration dialog provides the following options:

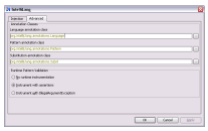
- Language injection for XML text
- Language injection for XML attributes
- Language injection for method parameters
- Advanced settings that let you change the names of the recognized base-annotations and the way how the pattern-validation is performed during runtime.

Screenshots

The screen shots below show some of the configuration options that can be tweaked and will be described in the following sections.







Language Injection

This tab allows you to configure the language injection feature for XML text, attributes and method parameters. Use the buttons in the toolbar or the context menu actions to add, remove, copy or import new entries. To add a new entry, the appropriate group has to be selected first.

Reordering the entries is possible with the Move Up and Move Down buttons. This can be important to define the precedence of different XML-injection entries. If an entry matches, no more injections are applied unless the injection specifies a value pattern .

XML Text

After adding a new XML text injection, select the ID of the Language to inject and optionally specify the prefix/suffix that make up the injection's context.

In the *XML Tag* pane, specify the local name (i.e. the name *without* any namespace prefix) and the namespace URI of the XML tag surrounding the text that should be treated as the selected language. The *name* field should not be empty, however the *namespace* field is optional.

The *Local Name* field takes a regular expression which makes it is possible to specify e.g. multiple tag names (name1|name2), case-insensitive names (e.g. (?i)tagname matches *tagname* as well as *TagName*), etc. Be sure not to enter any whitespace characters as they would be significant for the match.

XML Attributes

This is similar to the XML text configuration. However, it is possible to leave the *name* of the *XML Tag* empty which means that the configuration will apply to any attribute that matches the configured name, regardless of its containing XML tag.

The attribute's name takes the local name of the attribute and is also specified as a regular expression. This can be e.g. used to match e.g. HTML event handler attributes by specifying the name "on.*". The attribute name may also be empty (unless the tag name is empty as well) which means that the configuration applies to all attributes of the containing tag.

Advanced XML Options

The *Advanced* pane for XML Text and Attributes allows an even more fine-grained control over the injection process.

Value Pattern

This field takes a regular expression that determines what part of the XML text's or attribute's value the language should be injected into. This can be used to inject the language only into values that match a certain pattern or to inject it into multiple parts that match the pattern. This is done by using the first capturing group of the pattern as the target for the injection.

Examples:

`[$#]\{(.*?)\}`
This matches the pattern used by the JSP/JSF Expression Language.

`^javascript:(.*)`
This matches the *javascript*-protocol that can be used in hyperlink-hrefs to execute JS code.

XPath Condition

This field takes an XPath expression that can be used to address the injection-target more precisely than just by supplying its name and namespace URI. The context the expression is evaluated in is the surrounding XML tag for XML Text injection and the attribute itself for XML Attribute injection.

Example:

`lower-case(@type)='text/javascript'`
This limits the injection to tags whose type attribute contains the value "text/javascript".

It's possible to use the XPath extension functions that are provided by the Jaxen XPath engine, e.g. lower-case(). Also, there are three additional functions that can be used to determine the current file's name, extension and file type: file-name(), file-ext() and file-type(). You can also use normal code-completion to get a list of available functions.

Warning! For performance reasons, it's recommended to keep these expressions as simple as possible. Especially expressions that cause the whole document to be scanned, such as //foo/bar might cause performance problems with large files.

Method Parameters

This is a possibility to make use of IntelliLang's features, if, for any reason, the injection annotations cannot be used. This mainly applies to configuring 3rd party/library methods as well as projects that still have to use Java 1.4.

The language selection is identical to the one described above. To select one or more parameters of a certain method, first choose its containing class either by typing in the name (the textfield supports completion) or by using the class chooser available through the [...] button. Next, select a method using the [...] button inside the "Method-Name" pane (it's not possible to edit the method name manually). Once a method has been chosen, the table in the *Parameters* pane is populated with the parameters of the selected method. Select the check box in the first column to have the selected language injected into arguments passed for this parameter. Note that only parameters of type String can be selected.

There's already an IntelliJ IDEA-core (Inspection Gadgets rather) inspection that checks the arguments of some well-known methods to be a valid regular expression. However, IntelliLang can provide more detailed error messages and all the features of the Regular Expression Support plugin.

The XPath language that is configured by default for some of the standard XPath-APIs is provided by the XPathView + XSLT-Support plugin, which is available here .

Importing Configuration Entries

To import the injection configuration from another PhpStorm installation use the *Import* button from the toolbar and select the file "IntelliLang.xml" from /options or from a JAR file that contains some exported (via File | Export Settings) PhpStorm settings. After selecting the file, a new dialog displays the entries contained in that file. Use the Delete-button to remove all entries you don't want to import. Note that this will *not* change the selected configuration file.

This selective import-feature makes it easy to share certain configurations in a team without losing any local entries like when importing settings via the core File | Import Settings action.

To prevent inconsistent data, the import is only possible if the existing configuration is unchanged or has been saved with the *Apply* button.

Advanced Settings

The advanced configuration tab allows you to specify different names for the base-annotations that should be used. This helps to avoid any dependencies on foreign code where this is not desired or possible. The custom annotations should just provide the same properties as the original ones, i.e. value for all of them and an optional (default = "") prefix and suffix for the @Language replacement.

Here it's also possible to configure the kind of runtime checks the plugin should generate for the @Pattern validation:

- No instrumentation: No checks will be inserted and doesn't touch any compiled class files
- Instrumentation using assert: Pattern-validation is controlled with the -ea JVM switch and throws AssertionError. This is the recommended way due to the potentially negative impact on the performance for methods that are invoked very often.
- Instrumentation using IllegalArgument- (for method parameters) and IllegalStateExceptions (for return values). This is the same the @NotNull instrumentation of PhpStorm does.

Contributed Configurations

Additional configurations can be downloaded and imported from here. This community effect can help to minimize the configuration efforts and make IntelliLang even easier to use.

JavaScript-Specific Guidelines

PhpStorm enables creating rich Internet applications and Web applications by providing elaborate support of JavaScript and tight integration with AJAX and other adjacent frameworks and technologies.

In this section:

- JavaScript-Specific Guidelines
  - Prerequisites
  - JavaScript support
  - Developing an application that contains JavaScript
- Configuring JavaScript Libraries
- Creating JSDoc Comments
- Viewing JavaScript Reference
- Running and Debugging JavaScript
- JavaScript-Specific Refactorings
- Unit Testing JavaScript
- Minifying JavaScript
- Tracing with Spy-Js
- Using AngularJS
- Using Angular CLI
- Using Grunt Task Runner
- Using Gulp Task Runner
- Using JavaScript Code Quality Tools
- Using Meteor
- Using PhoneGap/Cordova

Prerequisites

Before you start working with JavaScript, make sure that the JavaScript Support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

JavaScript support

JavaScript files are marked with icon.

JavaScript support in PhpStorm includes:

- Switching between JavaScript language versions to choose the one that suits the targeted browser:
  - ECMAScript 3
  - ECMAScript 5.1
  - JavaScript 1.8.5
  - ECMAScript 6
  - ReactJS
  - Flow
- Full coding assistance:
  - Smart, DOM-based, browser-type aware JavaScript code completion for:
    - Keywords, labels, variables, parameters and functions.
    - User defined and built-in JavaScript functions.
    - JavaScript namespaces.
  - Error and syntax highlighting.
  - Code formatting and folding .
  - Numerous code inspections and quick-fixes , including the Switch to JavaScript <version> quick-fix.
  - Support of the strict mode standard.
  - Support for ECMAScript .
  - Initial support of ECMAScript Harmony .
- Code Generation
  - Generating code stubs based on file templates during file creation.
  - Inserting, expanding, and generating JavaScript code blocks using live templates .
  - Creating various applications elements via JavaScript and AJAX intention actions .
  - Possibility to create line and block comments (`Ctrl+Slash` / `Ctrl+Shift+Slash`).
  - Unwrapping and removing statements .
- Building type hierarchies and viewing type hierarchies .
- Refactoring

  PhpStorm provides both common refactoring types available for all the supported languages and JavaScript-specific refactoring.
  - Common refactorings :
    - Rename a file, function, variable, parameter, or label (both directly and via references).
    - Move/Copy a file.
    - Safe Delete a file.
    - Extract function.
  - JavaScript-Specific Extract Parameter Refactoring

- Numerous ways to navigate through the source code, among them:
  - Navigating with Structure Views .
  - Show/Goto Implementation (`Ctrl+Alt+B`) from overridden method / subclassed class.
- Advanced facilities for searching through the source code .
- Support of the JSDoc format and generating documentation comments .
- Viewing reference information:
  - Definitions, inline documentation, parameter hints.
  - JSDoc comments .
  - Lookup in external JavaScript libraries .
- Running and debugging.
  - Launching applications directly from PhpStorm by opening the starting application HTML page in the default PhpStorm browser .
  - A dedicated debug configuration for launching debugging sessions directly from PhpStorm.
  - A JavaScript-aware debugger that lets you execute applications step by step, evaluate expressions, examine related information and find runtime bugs.
  - Support for JavaScript breakpoints .
- Tight integration with related frameworks and technologies: AJAX , JQuery , YUI , DoJo , Prototype , MooTools , Qooxdoo , and Bindows :
  - Code completion for every framework.
  - DoJo style type annotations to provide more accurate completion and parameter type information.
  - Quick Documentation lookup for DoJo style commands.
- Support for the JSON (JavaScript Object Notation) format:
  - A JSON file type template mapped to `json` file extension.
  - JSON code formatting and folding .

To develop an application that contains JavaScript

Developing an application that contains JavaScript, generally, includes performing the following steps:

1. Make sure the JavaScript Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin . Make sure the JavaScript Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin . Make sure the JavaScript Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

2. Create a project to implement your application.

3. On the JavaScript page of the Settings dialog box, choose the JavaScript language version that suits the targeted browser.

4. Download, install, and configure JavaScript frameworks and libraries .

5. Populate the project . Use the following PhpStorm facilities, where applicable:
   - Coding assistance .
   - Code Generation .
   - Various types of navigation and search through the source code.
   - Viewing reference and generating documentation comments .
   - Look-up in external libraries .

6. Improve the quality and maintainability of your code using various types of refactoring , both common and JavaScript-specific.

7. Run your application by opening its starting HTML page in the PhpStorm default browser .

8. Debug your application.
   The JavaScript debugging functionality is incorporated in PhpStorm, so just configure the debugger , whereupon you can start the debugging session and proceed as usual : set the breakpoints, step through them, stop and resume the program, and examine it when suspended.

   Debugging for JavaScript applications is supported only in the Firefox and Google Chrome browsers.

Also in this part:

- Configuring JavaScript Libraries
- Creating JSDoc Comments
- Viewing JavaScript Reference
- Running and Debugging JavaScript
- JavaScript-Specific Refactorings
- Unit Testing JavaScript
- Minifying JavaScript
- Tracing with Spy-Js
- Using AngularJS
- Using Angular CLI
- Using Grunt Task Runner
- Using Gulp Task Runner
- Using JavaScript Code Quality Tools
- Using Meteor
- Using PhoneGap/Cordova

Configuring JavaScript Libraries

On this page:

- Basics
  - Predefined and custom libraries
  - Visibility and scope
- Viewing the libraries associated with a file
- Downloading and installing a JavaScript-related library from PhpStorm
- Configuring a custom JavaScript library
  - Removing a library file
  - Updating the contents of a library
  - Deleting a library
  - Specifying the scope to use a library in

Basics

When working in PhpStorm, you can use various JavaScript libraries and frameworks with full coding assistance at disposal. The only thing you need is "introduce" the required libraries to PhpStorm so they are recognized and references to them are successfully resolved.

Predefined and custom libraries

Right from PhpStorm, you can download and install a number of the most popular JavaScript libraries, such as: Dojo , ExtJS , jQuery , jQuery UI , Prototype , and others. Once installed, these libraries, by default, are visible in all files within the project.

You can also download any other JavaScript libraries and frameworks or even develop JavaScript libraries yourself, if necessary. To use such downloaded or self-developed libraries, you need to set them up as custom PhpStorm JavaScript libraries . Configured libraries can be used in code completion , highlighting, navigation , and Documentation Lookup .

To enable Documentation Lookup for symbols defined in an external library or framework, provide a link to its documentation. Upon pressing `Shift+F1` with the cursor positioned at the symbol in question, PhpStorm invokes this link and opens the documentation page in browser.

For jQuery , jQuery UI , Ext JS , Prototype , and Dojo library files, PhpStorm automatically detects links to the library documentation.

To get all the above mentioned coding assistance for a library or a framework, you need to configure it as a PhpStorm library and specify the library scope by associating the library with your project or its part, so you can reference the library from the files within this scope, get code completion, and retrieve definitions and documentation.

Please note, that configuring a framework as a PhpStorm library and associating it with a project only ensures coding assistance in the development environment, that is, while you are working in PhpStorm. To use a framework or library in the production environment, make sure the relevant version of the framework is available on the server.

Visibility and scope

Each PhpStorm library is characterized by its visibility status and scope.

The scope of a library defines the set of files and folders in which the library is considered as library, that is, it is write-protected, excluded from check for errors and refactoring, but only affects the completion list and highlighting.

The visibility status of a library determines whether it can be used in one project (Project) or can be re-used at the IDE level (Global).

- Once configured, a Global library can be associated with any of PhpStorm projects. The library itself can be located wherever you need, its settings are stored with other PhpStorm settings in the dedicated directories under the PhpStorm home directory.
  The advantage of configuring a framework as a global library is that you can store such library in one location and re-use it in unlimited number of your projects without copying the library under the project root every time.

  The disadvantage of this approach is that to enable team work on a project all the team members should have the library stored on their machines in the same location relative to the project root.
- A Project library is visible only within one single project. Therefore a project library can be associated only with this project or its part. This means that project libraries cannot be re-used, so if you later try to use a framework configured as a project library with another project, you will have to configure the library anew.
  The advantage of configuring a JavaScript framework as a project library is that you can share the library definition among other team members through the project settings so each developer does not have to configure the library separately,
- Predefined libraries bring JavaScript definitions (also known as “stubs”) for the standard DOM, HTML5 and EcmaScript API, as well as for Node.js global objects. These libraries make the basis for coding assistance in accordance with the API provided by the corresponding JavaScript engine. By enabling a certain predefined library you can ensure that your code fits the target environment.
  Predefined libraries are by default enabled in the scope of the entire project. A predefined library can be disabled or associated with another scope but it cannot be deleted .

Viewing the libraries associated with a file

1. Open the file in the editor.
2. Click the Hector icon on the Status bar. The pop-up window lists the libraries associates with the current file. To change the list, click the Libraries in scope links and edit the scope settings in the Manage Scope dialog box that opens.

Downloading and installing a JavaScript-related library from PhpStorm

PhpStorm provides a dedicated user interface for downloading and installing the most popular official JavaScript libraries. Using this interface, you can download and install Dojo , ExtJS , jQuery , jQuery UI , Prototype , and other libraries.

Besides the above listed official libraries, you can download stubs for TypeScript definition files .

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages and Frameworks node, and then click Libraries under JavaScript.
2. The JavaScript. Libraries page that opens shows a list of all the already available libraries. Click Download.

3. In the Download Library dialog that opens, choose the group of libraries in the drop-down list. The available options are Official libraries and TypeScript community stubs. Depending on your choice, PhpStorm displays a list of available libraries. Select the one to be downloaded and installed, and click Download and Install. You return to the JavaScript Libraries page where the new library is added to the list. CLick OK to save the settings.

Configuring a custom JavaScript library

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages and Frameworks node, and then click Libraries under JavaScript.
2. The JavaScript. Libraries page that opens shows a list of all the already available libraries. Click Add.
3. In the New Library dialog box that opens, specify the name of the library, the framework to configure the library from, and framework version to use.
4. Specify the library visibility :
   - To enable associating the library with the current project only, choose Current project.
   - To make the library available in any PhpStorm project, choose Global.
5. Create a list of files to be included in the library:
   1. Click the Add button + next to the list of library files and choose Attach Files or Attach Directory on the context menu, depending of whether you need separate files or an entire folder.
   2. Select the required file, files, or entire directory in the dialog box that opens. PhpStorm returns to the New Library dialog box where the Name read-only field shows the name of the selected library file or the names of relevant library files from the selected directory.
   3. In the Type field, specify which version of library you have downloaded and are going to add.
      - Choose Debug if you are adding a library file with uncompressed code. This version is helpful in the development environment, especially for debugging.
      - Choose Release if you are adding a library file with minified code. This version is helpful in the production environment because the file size is significantly smaller.

      It is recommended that you always have a debug version on hand along with the minified one. Minified code is hard to read and hard for PhpStorm to handle. When a debug version is available, PhpStorm automatically detects and ignores the minified file and retrieves definitions and documentation from the debug version.
   4. Specify the URL addresses to access the documentation for library files.
      - To add a link to the documentation for a library, select the corresponding library file, click the Specify button in the Documentation URLs area, and specify the documentation URL in the dialog box that opens.

        Tip For jQuery , jQuery UI , Ext JS , Prototype , and Dojo library files, PhpStorm automatically detects the link to the library documentation and suggests it in the Enter documentation URL text box.
      - To remove a link, select it in the Documentation URLs and click the Remove button.

Removing a library file

In the New Library / Edit Library dialog box, select the required library file and click the Remove button — .

Updating the contents of a library

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages and Frameworks node, and then click Libraries under JavaScript.
2. The JavaScript. Libraries page that opens shows a list of all the already available libraries. Select the required library and click Edit.
3. In the Edit Library dialog box that opens, add library files, remove library files, and change links to documentation as necessary.

Deleting a library

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages and Frameworks node, and then click Libraries under JavaScript.
2. The JavaScript. Libraries page that opens shows a list of all the already available libraries. Select the required library and click Remove.

Specifying the scope to use a library in

The scope of a library defines the set of files and folders in which the library is considered as library, that is, it is write-protected, excluded from check for errors and refactoring, but only affects the completion list and highlighting.

By default, all predefined libraries and libraries downloaded from PhpStorm provide completion, resolution, highlighting and are treated as libraries in any file within the project. In other words, their usage scope is the whole project.

Libraries that you create yourself are not considered libraries in any of the files unless you specify their usage scope explicitly.

To specify the scope for a library:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages and Frameworks node, and then click Libraries under JavaScript.
2. The JavaScript. Libraries page that opens shows a list of all the already available libraries. Click Manage Scopes.
3. In the JavaScript Libraries. Usage Scope dialog box that opens, specify the custom JavaScript libraries to use in files and folders within your project. To appoint a library for a file or folder, select the required item in the File/Directory field and choose the relevant library from the Library drop-down list.
   The contents of the list depend on the visibility type of the configured libraries. Global libraries are on the list in all PhpStorm projects. Project libraries are on the list only within the project they were originally configured in.

Creating JSDoc Comments

To make your code easy to use by other developers it is considered good practice to provide an HTML documentation of its Application Programming Interface (API). Such documentation can be generated automatically by the JSDoc tool. All you need, is supply your code with documentation comments in accordance with the JSDoc standard. The tool retrieves information from your comments and renders it in HTML using a built-in template.

You can find a detailed description of the JSDoc syntax with examples and explanation of their use in the article An Introduction to JSDoc .

PhpStorm creates stubs of JSDoc comments on typing the opening tag `/**` and pressing `Enter`. If this feature is applied to a method or a function , `@param` tags are created. In any other places PhpStorm adds an empty documentation stub.

TODO patterns and Closure Compiler annotations inside documentation comments are also recognized and are involved in code completion, intention actions, and other types of coding assistance.

Documentation comments in your source code are available for the Quick Documentation Lookup and open for review on pressing `Ctrl+Q`.

PhpStorm checks syntax in the comments and treats it according to the Code Inspections settings.

On this page:

– Example of JavaScript comment
– Enabling generation of documentation comments
– Creating a JSDoc comment block

Example of JavaScript comment

Consider the following function:

```
function loadDocs(myParam1, myParam2){}
```

Type the opening documentation comment and press `Enter` to generate the documentation comment stub:

```
/**
* @param myParam1
* @param myParam2
*/
```

To enable or disable generation of documentation comments

1. Open the Settings dialog box , expand the Editor node, then expand General node, and click the Smart Keys page.
2. In the Enter section, select or clear Insert documentation comment stub check box.

To create a JSDoc comment block for a method or a function

1. Place the caret before the method or function declaration.
2. Type the opening block comment `/**` and press `Enter`.
3. Describe the listed parameters and return values.

Documentation comment can be created with the dedicated action Fix Doc Comment. It can be invoked by means of Find Action command.

Press `Ctrl+Shift+A`, with the caret somewhere within a class, method, function, or field, which should be documented, and enter the action name Fix Doc String. The missing documentation stub with the corresponding tags is added. For example:

```
function loadDocs(myParam1, myParam2){}
```

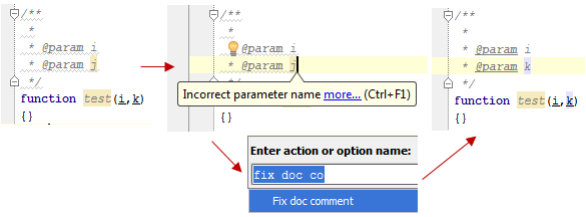
Type the opening documentation comment and press `Enter` to generate the documentation comment stub:

```
/**
* @param myParam1
* @param myParam2
*/
```

The next case lays with fixing problems in the existing documentation comments.

For example, if a method signature has been changed, PhpStorm highlights a tag that doesn't match the method signature.

For JavaScript, PhpStorm suggests an intention action UpdateJSDoc comment. You can also press `Ctrl+Shift+A`, and type the action name:



Tip The action Fix doc comment has no keyboard shortcut bound with it. You can configure keyboard shortcut of your own.

Viewing JavaScript Reference

Besides common referential procedures that are available in the context of any supported language, PhpStorm provides the following ways of retrieving JavaScript-specific reference:

- Viewing Inline Documentation
- Documentation Look-up in External JavaScript Libraries

Viewing Inline Documentation

Quick Documentation Lookup helps you get quick information for any symbol, provided that this symbol has been supplied with Documentation comments in the applicable format. PhpStorm recognizes inline documentation created in accordance with the JavaScript Documentation Tool format.

On this page:

– Viewing documentation for a symbol at caret
– Changing font style of the quicj documentation
– Toolbar of the quick documentation lookup

To view documentation for a symbol at caret, do one of the following

– On the main menu, choose View | Quick Documentation Lookup.

– Press `Ctrl+Q`.

– Provided that the check box Show quick doc on mouse move in the editor settings is selected, just move your mouse pointer over the desired symbol.

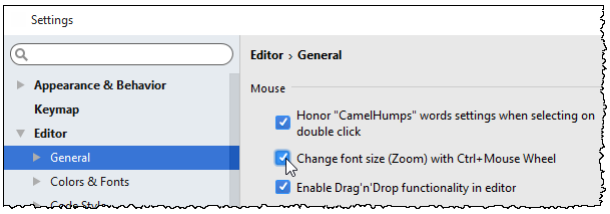
When you explicitly invoke code completion, then quick documentation for an entry selected in the suggestion list can be displayed automatically. The behavior of quick documentation lookup is configured in the Code Completion page of the Settings/Preferences dialog.

Tip Sequential pressing `Ctrl+Q` toggles the focus of the Quick Documentation pop-up window and Documentation Tool Window.

To change the font size of quick documentation, do one of the following

– Click in the upper-right corner of the quick documentation window, and move the slider.

– Rotate the mouse wheel while keeping the `Ctrl` key pressed.
Note that for this feature to work, you have to enable it in the General page of the editor settings.



Refer to the section Zooming in the Editor for details.

Toolbar of the quick documentation lookup

The Quick Documentation Lookup window helps navigate to the related symbols via hyperlinks, and provides a toolbar for moving back and forth through the already navigated pages, changing font size, and viewing documentation in an external browser.

When pinned, the Quick Documentation Lookup turns into Documentation Tool Window, with the corresponding sidebar icon, and more controls.

Icon	Shortcut	Description
	`Left` or `Right`	Switch to the previous or next documentation page (e.g. after using hyperlinks). Note On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.
	`Shift+F1`	View external documentation in the default browser.
	`F4`	Switch to the item (e.g. source) that corresponds to the documentation page currently shown.
		Turn the Auto-update from source option on or off. When the option is on, the information in the tool window is synchronized with your navigation in the editor and other places in the UI.
		Click this icon to show font size slider. Move the slider to increase or decrease the font size in the quick documentation window as required.

Documentation Look-up in External JavaScript Libraries

Besides common referential procedures and viewing JSDoc comments , you can also retrieve documentation for symbols defined in external JavaScript libraries and frameworks.

When this functionality is enabled, upon pressing `Shift+F1` with the cursor positioned at the symbol in question, PhpStorm invokes the corresponding link and opens the documentation page in the default PhpStorm browser .

- The default PhpStorm browser is configured on the Web Browsers page of the Settings dialog.
- For more details on specifying the browser to use by default, see Configuring Browsers .

To enable the external JavaScript library lookup functionality

1. Download the required libraries or frameworks.
2. Configure the downloads as libraries at the PhpStorm level.
3. Specify links to external documentation .

To view documentation on a symbol defined in an external JavaScript library

- Position the cursor at the symbol in question and choose View | External Documentation, or press `Shift+F1`.

Running and Debugging JavaScript

On this page:

- Running JavaScript
- Debugging JavaScript

To run a file with injected JavaScript from PhpStorm

No run configuration is required for launching applications with injected JavaScript from PhpStorm. You just open the HTML page with the reference to the required script in the browser and that's it.

1. In the editor, open the HTML file with the JavaScript reference. This HTML file does not necessarily have to be the one that implements the starting page of the application.

2. Do one of the following:
   - Choose View | Preview file in on the main menu or press `Alt+F2`. Then select the desired browser from the pop-up menu.
   - Hover your mouse pointer over the code to show the browser icons bar: Click the icon that indicates the desired browser.

To debug JavaScript code, perform these general steps:

1. Configure the JavaScript debugger .

2. Configure and set breakpoints in the JavaScript code as required.

3. Initiate a debugging session .

4. If you are using Chrome or Dartium, install and configure the JetBrains Chrome Extension. For details, see Using JetBrains Chrome Extension .

5. Step through the program , stop and resume program execution, examine it when suspended , etc.

Configuring JavaScript Debugger

On this page:

- Introduction
- Common JavaScript debugger options

Introduction

Debugging JavaScript in PhpStorm is supported through the JavaScript Debugger. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Debugging of JavaScript code is supported in Google Chrome and other browsers of the Chrome by means of browser-specific JetBrains extensions. No special steps are required from your side to install or enable these extensions. PhpStorm does it for when you initiate a debugging session for the first time.

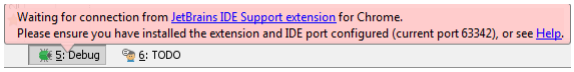
During a debugging session, PhpStorm considers any injected JavaScript code in a Web page opened in the browser as subject for debugging. This means that you cannot use the same instance of a browser for browsing and for debugging simultaneously. For Google Chrome you can solve this problem by configuring PhpStorm to use a separate Chrome user profile for debugging.

Common JavaScript debugger options

To configure the settings for debugging JavaScript code

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Build, Execution, Deployment node, and then click Data Views under Debugger.

2. On the Data Views page that opens, specify the following:
   - Use the Built-in server port spin-box to specify the number of the port where the built-in web server runs. The port number should be set to 1024 or higher. By default this port is set to the default PhpStorm port through which PhpStorm accepts connections from services (`63342`). During a debugging session, the Chrome extension will listen to this port. If for some reason the default PhpStorm port is already busy, PhpStorm finds the closest available port and starts on it. This results in a conflict: PhpStorm is running on a "new" port while the Chrome extension still listens to the port of a previously started product and fails to establish connection with PhpStorm. The conflict reveals when you initiate a debugging session from PhpStorm: the extension fails to connect through the default port, PhpStorm waits for connection from the extension and displays the following message with the port number where it is actually running:

     To fix the problem, specify the actual port in the Chrome extension options, see Using JetBrains Chrome Extension .
   - In the Value tooltip delay (ms) field , specify the time period between pointing at an object and displaying the object's value in a tooltip.
   - Whether to display DOM properties in a tab or not.
   - Whether to display function values in a tab or not. If you want the function values to be shown only for user-defined functions, select the Show only user-defined functions. (Otherwise, the function values will be shown for all function.)
   - Whether you want object node properties to be shown. If so, specify the properties. Use + and — to manage the list of properties.
   - Whether you want the debugger to skip certain scripts. If so, specify URLs for the corresponding scripts. Use + and — to manage the list of script URLs.

Debugging JavaScript

On this page:

- Basic terms
- Starting a debugging session by opening an HTML file from PhpStorm (local debugging)
- Starting a debugging session through a user-defined configuration (local debugging)
- Local debugging in Firefox
- Remote debugging
- Mapping
- Debugging asynchronous code
- Debugging workers

Basic terms

When an HTML file with injected JavaScript code is opened in the browser, the script execution starts and suspends on reaching the first breakpoint, whereupon you can analyze the suspended program, step through it, resume execution when ready, etc. This HTML file does not necessarily have to be the one that implements the starting page of the application.

Depending on the location of the application files, PhpStorm distinguishes between local and remote debugging modes.

1. Local debugging. In this mode, the application files remain where they are in the PhpStorm project on you machine. PhpStorm runs them on the built-in Web server with the root URL `http://localhost:<built-in server port>/<project root>` . The built-in server port (1024 or higher) is specified on the Data Views page of the Settings dialog box. A debugging session in this mode can be initiated in two ways:
   - Open the HTML file with the JavaScript injection to be tested and launch the temporary run/debug configuration that PhpStorm has generated automatically. This approach it helpful when you do not need to debug the entire application but just one script.
   - Create and launch a permanent debug configuration.
2. Remote debugging. In this mode, the application files are on an external web server and you have their copies in a PhpStorm project on your computer. No matter, whether the web server itself is running on a physically remote host or on your machine, application files deployed on it are treated as remote. For details, see Working with Web Servers: Copying Files .
   You can deploy the application files to a web server and launch the application yourself or debug an already running application.

   When a remote HTML file with a JavaScript injection is opened, the debugger tells PhpStorm the name of the currently processed file and the number of the line to be processed. PhpStorm opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called mapping, it is set in the debug configuration.

   A Remote debugging session can be initiated only through a permanent debug configuration that you have to define yourself manually.

PhpStorm supports local and remote JavaScript debugging only in Google Chrome and other browsers of the Chrome family. In Firefox , remote debugging is not supported at all. Local debugging is supported for Firefox, version 36 an higher, through the Firefox Remote debug configuration.

Starting a debugging session by opening an HTML file from PhpStorm (local debugging)

1. Set the breakpoints in the JavaScript code, as required.
2. Open the HTML file with the JavaScript injection to debug or select the HTML file in the Project view .
3. On the context menu of the editor or the selection, choose Debug <file name>. The automatically generated debug configuration starts a debugging session. This configuration is temporary but you can save it later.
   The file opens in the default browser and the Debug tool window appears.
4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the Debug tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension .
5. In the Debug tool window, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , view actual HTML DOM , edit the code , etc.

Starting a debugging session through a user-defined configuration (local debugging)

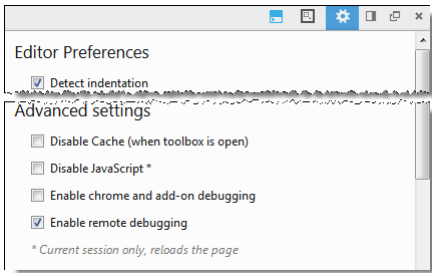
1. Set the breakpoints in the JavaScript code, as required.
2. Create a permanent debug configuration.
   1. On the main menu, choose Run | Edit Configurations.
   2. Click the Add New Configuration toolbar button +, and choose JavaScript Debug on the context menu.
   3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, specify the following:
      - The URL address of the HTML file that implements the page to start debugging from. Use the following format: `http://localhost:<built-in server port>/<project root>/<path to the HTML file relative to the project root>`
      - Choose the browser to debug the application in (Firefox or Chrome).
   4. Click OK to save the configuration settings.
3. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button .
   The HTML file specified in the run configuration opens in the chosen browser and the Debug tool window appears.
4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the Debug tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension .
5. In the Debug tool window, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , view actual HTML DOM , edit the code , etc.

Local debugging in Firefox

In Firefox , remote debugging is not supported at all. Local debugging is supported for Firefox, version 36 an higher, through the Firefox Remote debug configuration.

1. Enable debugging in Firefox:
   1. Open your Firefox browser, then open Tools | Web Developer | Toggle Tools.
   2. In the Development Tools pane that opens, click Toolbox Options button on the toolbar and select the Enable remote debugging check box under Advanced

Settings.



3. In the console at the bottom of the browser (the console appeared opened when you opened the Developer toolbar), type `listen <port number>`.

   ```
   » listen 6000
   ```

   You can specify any port number, however it is recommended that you use 6000 and higher. Later you will specify this port number in the run configuration.

If you completed the above mentioned steps successfully, a pop-up window with the following message is shown: Listening on port 6000.

2. Create a run/debug configuration:
   1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
   2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button, and choose Firefox Remote on the context menu. Do not feel mislead by the term Remote. In this context it reflects the technical details of the implementation.
   3. In the Run/Debug Configuration: Firefox Remote dialog box that opens, specify the host where the application is running (currently it is always `localhost`) and the port the debugger will listen to. It must be the port that you specified when enabling debugging in Firefox. The default value is 6000.
   4. Click OK to save the configuration settings.
3. Set the breakpoints where necessary.
4. Open the HTML file with the related JavaScript code the editor or select it in the Project tool window, choose Open in Browser on the context menu of the selection, then choose Firefox. The browser opens showing the application after the code execution, that is, the breakpoints you set have no effect yet.
5. Choose the newly created configuration in the Select run/debug configuration drop-down list on the toolbar and click the Debug toolbar button. In the dialog bo that opens, click OK to allow incoming connections. From the Choose Page to Debug pop-up list, choose the page with your application.
6. Refresh the page with your application in Firefox: the page shows the results of code execution up to the first breakpoint.
7. In the Debug tool window, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , view actual HTML DOM , edit the code , etc.

Remote debugging

1. Set the breakpoints in the JavaScript code, as required.
2. Create a permanent debug configuration.
   1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
   2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button, and choose JavaScript Debug on the context menu.
   3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, specify the following:
      - The URL address of the HTML file that implements the page to start debugging from. This URL address should correspond with the server access configuration.
      - Choose the browser to debug the application in (Chrome or another browser of the Chrome family).
      - Optionally define mappings between local files and URL addresses of their copies on the server. These mappings are required only if the local folder structure under the project root differs from the folder structure on the server. If the structures are identical, PhpStorm itself "retrieves" the paths to local files by parsing the URL addresses of their copies on the server.

      Click OK to save the configuration settings.
3. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button.
   The HTML file specified in the run configuration opens in the chosen browser and the Debug tool window appears.
4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the Debug tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension .
5. In the Debug tool window, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , view actual HTML DOM , edit the code , etc.

Mapping

If you are using a third-party Web server (not the built-in PhpStorm server) for debugging or you have defined several resource root folders in addition to the project root, you need to map files, folders, or the entire project folder to the files or folders on the server. When a remote HTML file with a JavaScript injection is opened, the debugger tells PhpStorm the name of the currently processed file and the number of the line to be processed. PhpStorm opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called mapping, it is set in the debug configuration.

To configure mappings:

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button, and choose JavaScript Debug on the context menu.
3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, map the files or folders or the entire project folder to URL addresses of files and folders on the server.

Debugging asynchronous code

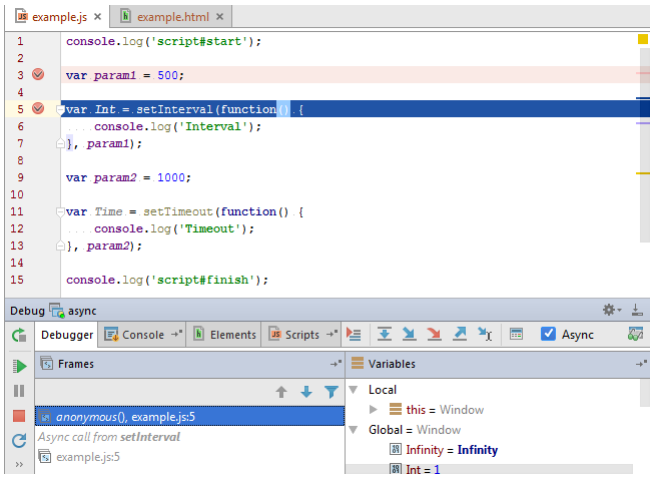
PhpStorm supports debugging asynchronous client-side JavaScript code, currently this functionality is available only in Chrome. The asynchronous debugging mode is toggled through the Async check box on the tool bar of the Debugger tab in the `Debug` tool window. The check box is displayed only during a JavaScript debugging session.

- When this check box is selected, PhpStorm recognizes breakpoints inside asynchronous code and stops at them and lets you step into asynchronous code. As soon as a breakpoint inside an asynchronous function is hit or you step into asynchronous code, a new element `Async call from <caller>` is added in the Frames pane of the

Debugger tab. PhpStorm displays a full call stack, including the caller and the entire way to the beginning of the asynchronous actions.

- When the check box is cleared, PhpStorm does not recognize and therefore skips breakpoints in the asynchronous code and does not allow you to step into it.

The image below shows an example of a JavaScript debugging session.



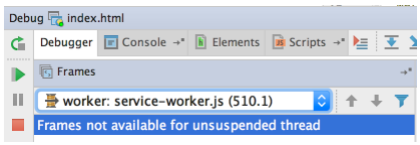
- With the Async check box selected, the debugger will stop at line3(breakpont), then at line5(breakpoint). On clicking Step into, the debugger will stop at line5 (on `function`), then will move to line6.
- With the Async check box cleared, the debugger will stop at line3(breakpont), then at line5(breakpoint). On clicking Step into, the debugger will move to line9.

Debugging workers

PhpStorm supports debugging Service Workers and Web Workers . PhpStorm recognizes breakpoints in each worker and shows the debug data for it as a separate thread in the Frame pane on the Debugger tab of the Debug Tool Window .

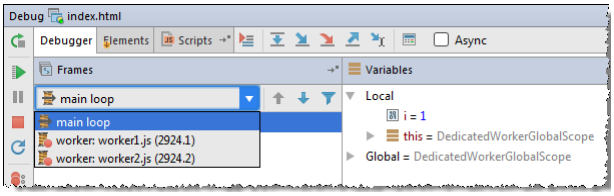
Note that PhpStorm can debug only dedicated workers , debugging for shared workers is currently no supported.

1. Set the breakpoints in the Workers to debug.
2. If you are using Service Workers, make sure the Allow unsigned requests check box on the Debugger page is selected. Otherwise your service workers may be unavailable during a debug session:



3. Create a JavaScript Debug run/debug configuration:
   1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
   2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button +, and choose JavaScript Debug on the context menu.
   3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, specify the following:
      - The URL address of the HTML file that implements the page to start debugging from. This URL address should correspond with the server access configuration.
      - Choose the browser to debug the application in (Chrome or another browser of the Chrome family).
      - Optionally define mappings between local files and URL addresses of their copies on the server. These mappings are required only if the local folder structure under the project root differs from the folder structure on the server. If the structures are identical, PhpStorm itself "retrieves" the paths to local files by parsing the URL addresses of their copies on the server.

   Click OK to save the configuration settings.
4. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button.
   The HTML file specified in the run configuration opens in the chosen browser and the Debug tool window opens with the Frames drop-down list showing all the Workers:



To examine the data (variables, watches, etc.) for a Worker, select its thread in the list and view its data in the Variables and Debug Tool Window. Watches panes. When you select another Worker, the contents of the panes are updated accordingly.

Setting Labels to Variables, Objects and Watches

On this page:

– Introduction

– Settings and removing labels

Introduction

You can add your own label to a variable, object, or watch and then reference it as if it were a local variable `<label-name>_DebugLabel` defined in the same context where the expression is evaluated. PhpStorm also displays these labels in suggestion pop-up lists for code completion in the Evaluate Expression dialog box.

Settings and removing labels

To set a label

1. Select the desired watch in the list
2. Select Mark Object on the context menu or press `F11`. The Select object label dialog box opens.
3. Specify the label name.
4. Click the Browse button ... to change the label color. Click OK when ready.

To remove a label, right-click the item and select Unmark Object on the context menu, or select the item in the list and press `F11`.

JavaScript-Specific Refactorings

In addition to the full range of common refactorings , PhpStorm provides the following JavaScript-specific refactorings:

- Extract Parameter in JavaScript
- Change Signature in JavaScript
- Extract Variable in JavaScript

Extract Parameter in JavaScript

The Extract Parameter refactoring is used to add a new parameter to a method declaration and to update the method calls accordingly.

– Examples

– Extracting a parameter in JavaScript

Examples

The following table shows two different ways of extracting a function parameter.

In the first of the examples a new parameter is extracted as an optional parameter. So the corresponding function call doesn't change.

In the second of the examples the parameter is extracted as a required parameter. So the corresponding function call changes accordingly.

Before After

```
// A new parameter will be added to this
                    // function to replace the 1's:

                    function calculate_sum(i) {

                    alert('Adding ' + 1 + ' to ' + i);
                    return  1 + i;
                    }

                    function show_sum() {

                    // Here is the function call:

                    alert('Result: ' + calculate_sum(5));
                    }

                    // When adding a new parameter we'll specify
                    // that it should be an optional one.
```

```
// The new parameter i2 has been added
                    // as an optional parameter:

                    function calculate_sum(i, i2) {
                    i2 = i2 || 1;
                    alert('Adding ' + i2 + ' to ' + i);
                    return i2 + i;
                    }

                    function show_sum() {

                    // The function call has not changed:

                    alert('Result: ' + calculate_sum(5));
                    }
```

```
// A new parameter will be added to this
                    // function to replace the 1's:

                    function calculate_sum(i) {
                    alert('Adding ' + 1 + ' to ' + i);
                    return 1 + i;
                    }

                    function show_sum() {

                    // Here is the function call:

                    alert('Result: ' + calculate_sum(5));
                    }

                    // When adding a new parameter we'll specify
                    // that it should be a required one.
```

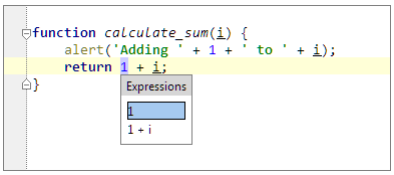
```
// The new parameter i2 has been added
                    // as a required parameter:

                    function calculate_sum(i, i2) {
                    alert('Adding ' + i2 + ' to ' + i);
                    return i2 + i;
                    }

                    function show_sum() {

                    // The function call changed accordingly:

                    alert('Result: ' + calculate_sum(5, 1));
                    }
```

Extracting a parameter in JavaScript

1. In the editor, place the cursor within the expression to be replaced by a parameter.

2. Do one of the following:
   – Press `Ctrl+Alt+P`.
   – Choose Refactor | Extract | Parameter on the main menu.
   – Choose Refactor | Extract | Parameter from the context menu.

3. If more than one expression is detected for the current cursor position, the Expressions list appears. If this is the case, select the required expression. To do that, click the expression. Alternatively, use the `Up` and `Down` arrow keys to navigate to the expression of interest, and then press `Enter` to select it.



4. In the Extract Parameter dialog :
   1. Specify the parameter name in the Name field.
   2. The Value field, initially, contains the expression that you have selected. Normally, you don't need to change this. (Depending on whether the option Optional parameter is selected or not, this will be the value assigned to the new parameter in the function body or passed to the function in the function calls.)
   3. If, when extracting a new parameter, you don't want to change the function calls, select the Optional parameter check box. If you do so, the value specified in the Value field will be assigned to the new parameter in the function body. The calls to the function (if any) won't change.
      If you want to pass the value (specified in the Value field) to the new parameter through the existing function calls, clear the Optional parameter check box.

If you do so, all the function calls will change according to the new function signature; no explicit assignment of the parameter value will be added to the function body.

4. If more than one occurrence of the expression is found within the function body, you can choose to replace only the selected occurrence or all the found occurrences with the references to the new parameter. Use the Replace all occurrences check box to specify your intention.
5. Click OK.

```
function calculate_sum(i, i2) {
    i2 = i2 || 1;
    alert('Adding ' + i2 + ' to ' + i);
    return i2 + i;
}
```

Change Signature in JavaScript

On this page:

- Introduction
- Examples
- Changing function signature

Introduction

In JavaScript, you can use the Change Signature refactoring to:

- Change the function name.
- Add new parameters and remove the existing ones. Note that you can also add a parameter using a dedicated Extract Parameter refactoring.
- Reorder parameters.
- Change parameter names.
- Propagate new parameters through the method call hierarchy.

Examples

The following table shows 4 different ways of performing the same Change Signature refactoring.

In all the cases, the function `result()` is renamed to `generate_result()` and a new parameter `input` is added to this function.

The examples show how the function call, the calling function (`show_result()`) and other code fragments may be affected depending on the refactoring settings.

Before After

```
// This is the function whose signature will be changed:

function$#160;result() {
// some code here
}

function show_result() {

// Here is the function call:

alert('Result: ' + result());
}

// Now we'll rename the function and
// add one (required) parameter.
```

```
// The function has been renamed to generate_result.
// The new parameter input has been added.

function generate_result(input) {
// some code here
}

function show_result() {

// The function call changed accordingly:

alert('Result:
' + generate_result(100));
}

// When performing the refactoring, 100 was specified as
// the parameter value.
```

```
// This is the function whose signature will be changed:

function result() {

// some code here
}

function show_result() {

// Here is the function call:

alert('Result: ' + result());
}

// Now we'll rename the function and add one parameter.
// This time, we'll specify that the parameter is optional.
```

```
// The function has been renamed to generate_result.
// The new optional parameter input has been added.

function generate_result(input) {
input = input || 100;
// some code here
}

function show_result() {

// The function call changed accordingly:

alert('Result: ' + generate_result(100));
}

// When performing the refactoring, 100 was specified as
// the parameter value.
```

```
// This is the function whose signature will be changed:

function result() {
// some code here
}

// This function will also change its signature:

function show_result() {

// Here is the function call:

alert('Result: ' + result());
}

// Now we'll rename the function and add one required
// parameter. We'll also ask PhpStorm to propagate
// the new parameter through the calling function show_result()
// to the function call.
```

```
// The function has been renamed to generate_result.
// The new parameter input has been added.

function generate_result(input) {
// some code here
}

// Note the new function parameter:

function show_result(input) {

// The function call changed accordingly:

alert('Result: ' + generate_result(input));
}
```

```
// This is the function whose signature will be changed:
function result() {

// some code here
}

// This function will also change its signature:

function show_result() {

// Here is the function call:

alert('Result: ' + result());
}

// Now we'll rename the function and add one optional
// parameter. We'll also ask PhpStorm to propagate
// the new parameter through the calling function show_result()
// to the function call.
```

```
// The function has been renamed to generate_result.
// The new optional parameter input has been added.

function generate_result(input) {
input = input || 100;
// some code here
}

// Note the new function parameter:

function show_result(input) {
input = input || 100;

// The function call changed accordingly:

alert('Result: ' + generate_result(input));
}

// When performing the refactoring, 100 was specified as
// the parameter value.
```

To change a function signature

1. In the editor, place the cursor within the name of the function whose signature you want to change.

2. Do one of the following:
   - Press `Ctrl+F6`.
   - Choose Refactor | Change Signature in the main menu.
   - Select Refactor | Change Signature from the context menu.

3. In the Change Signature dialog , make the necessary changes to the function signature and specify which other, related changes are required.
   You can:
   - Change the function name. To do that, edit the text in the Name field.
   - Manage the function parameters using the table of parameters and the buttons to the right of it:
     - To add a new parameter, click + (`Alt+Insert`) and specify the properties of the new parameter in the corresponding fields.
       When adding parameters, you may want to propagate these parameters to the functions that call the current function.
     - To remove a parameter, click any of the cells in the corresponding row and click — (`Alt+Delete`).
     - To reorder the parameters, use ↑ (`Alt+Up`) and ↓ (`Alt+Down`). For example, if you wanted to make a certain parameter the first in the list, you would click any of the cells in the row corresponding to that parameter, and then click ↑ the required number of times.
     - To change the name of a parameter, make the necessary edits in the corresponding table cell.
   - Propagate new function parameters (if any) along the hierarchy of the functions that call the current function.
     (There may be the functions that call the function whose signature you are changing. These functions, in their turn, may be called by other functions, and so on. You can propagate the changes you are making to the parameters of the current function through the hierarchy of the calling functions and also specify which calling functions should be affected and which shouldn't.)

     To propagate the new parameters:
     1. Click .
     2. In the left-hand pane of the Select Methods to Propagate New Parameters dialog, expand the necessary nodes and select the check boxes next to the functions you want the new parameters to be propagated to.
        To help you select the necessary functions, the code for the calling function and the function being called is shown in the right-hand part of the dialog (in the Caller Method and Callee Method panes respectively).

        As you switch between the functions in the left-hand pane, the code in the right-hand part changes accordingly.
     3. Click OK.

4. To perform the refactoring right away, click Refactor.
   To see the expected changes and make the necessary adjustments prior to actually performing the refactoring, click Preview.

Extract Variable in JavaScript

In JavaScript and ActionScript, you can replace an expression with a variable or a constant. For JavaScript 1.7 or a later version, there is also an option of extracting a local variable.

To perform this refactoring, you can use:

- In-place refactoring . In this case you specify the new name right in the editor.
- Refactoring dialog , where you specify all the required information. To make such a dialog accessible, you have to clear the check box Enable in-place mode in the editor settings.

JavaScript examples

Before After

```
Parenizor.method('toString', function ()
                    {
                    return '(' + this.getValue() + ')';
                    }
```

GLOBAL VARIABLE

```
Parenizor.method('toString', function () {
                    var string = '(' + this.getValue()
                    + ')';
                    return string;
                    }
```

LOCAL VARIABLE

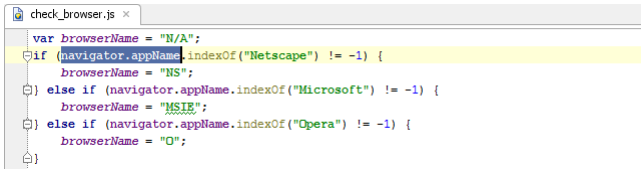
```
Parenizor.method('toString', function () {
                    let string = '(' + this.getValue()
                    + ')';
                    return string;
                    }
```

```
var browserName =
                    "N/A";
                    if (navigator.appName.indexOf("Netscape") != -1) {
                    browserName = "NS";
                    } else if (navigator.appName.indexOf("Microsoft") != -1) {
                    browserName = "MSIE";
                    } else if (navigator.appName.indexOf("Opera") != -1) {
                    browserName = "O";
                    }
```

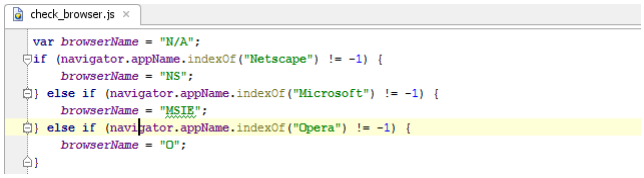
```
var browserName =
                    "N/A";
                    var appName = navigator.appName;
                    if (appName.indexOf("Netscape") != -1) {
                    browserName = "NS";
                    } else if (appName.indexOf("Microsoft") != -1) {
                    browserName = "MSIE";
                    } else if (appName.indexOf("Opera")
                    != -1) {
                    browserName = "O";
                    }
```

To extract a variable using in-place refactoring

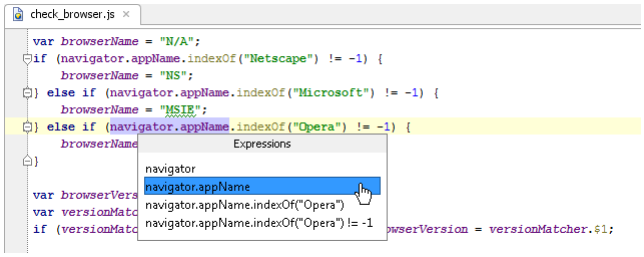
1. In the editor, select the expression to be replaced with a variable. You can do that yourself or use the smart expression selection feature to let PhpStorm help you. So, do one of the following:
   - Highlight the expression. Then choose Refactor | Extract | Variable on the main menu or on the context menu.
     Alternatively, press `Ctrl+Alt+V`.



   - Place the cursor before or within the expression. Choose Refactor | Extract Variable on the main menu or on the context menu. or press `Ctrl+Alt+V`.



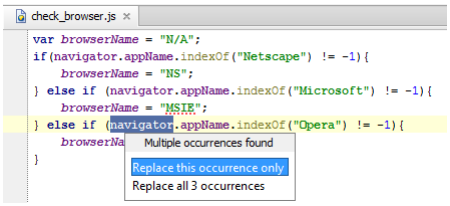
     In the Expressions pop-up menu, select the expression. To do that, click the required expression. Alternatively, use the Up and Down arrow keys to navigate to the expression of interest, and then press `Enter` to select it.



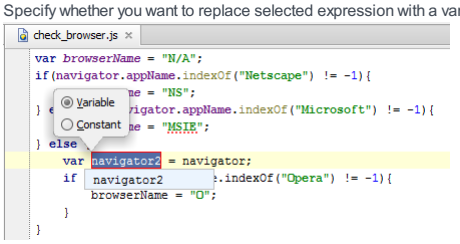
Note The Expressions pop-up menu contains all the expressions appropriate for the current cursor position in the editor.

When you navigate through the suggested expressions in the pop-up, the code highlighting in the editor changes accordingly.

2. If more than one occurrence of the selected expression is found, select Replace this occurrence only or Replace all occurrences in the Multiple occurrences found pop-up menu. To select the required option, just click it. Alternatively, use the `Up` and `Down` arrow keys to navigate to the option of interest, and press `Enter` to select it.



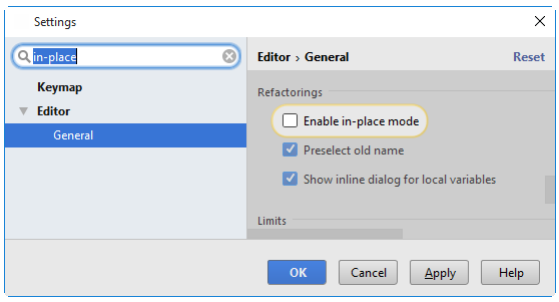
3. Specify whether you want to replace selected expression with a variable, or a constant. To do that, click the desired radio button in the pop-up balloon:



4. Specify the name of the variable. Do one of the following:
   - Select one of the suggested names from the pop-up list. To do that, double-click the suitable name. Alternatively, use the `Up` and `Down` arrow keys to navigate to the name of interest, and `Enter` to select it. When finished, press `Escape`.
   - Edit the name by typing. The name is shown in the box with red borders and changes as you type. When finished, press `Escape`.

To extract a variable using the dialog box

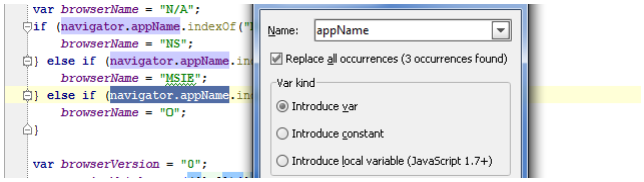
If the Enable in place refactorings check box is cleared in the Editor settings, the Extract Variable refactoring is performed by means of the dialog box.



1. Select the desired expression, and invoke Extract Variable refactoring as described above .

2. If more than one expression is detected for the current cursor position, the Expressions list appears. If this is the case, select the required expression. To do that, click the expression. Alternatively, use the `Up` and `Down` arrow keys to navigate to the expression of interest, and then press `Enter` to select it.

3. In the Extract Variable Dialog :
   - Specify the variable name next to Name. You can select one of the suggested names from the list or type the name in the Name box.
   - If more than one occurrence of the selected expression is found, you can select to replace all the found occurrences by selecting the corresponding check box. If you want to replace only the current occurrence, clear the Replace all occurrences check box.
   - Specify the type of variable to be extracted: a (global) variable, a constant, or a local variable. To do that, click the required option in the Var kind area.
   - For ActionScript, you can choose to introduce a constant rather than a variable. To do that, select the Make constant check box.
   - Click OK.



Unit Testing JavaScript

PhpStorm provides facilities for running JavaScript unit tests against a local or remote test server. You can run test cases, methods, or entire test files.

To create and run unit tests in JavaScript, perform these general steps

1. Enable JavaScript unit testing support .
2. Write the unit tests .
3. To have PhpStorm recognize the unit tests and detect the corresponding production source code, mark the folder where the unit tests are stored as test folder .
4. If applicable, write configuration files that define which test and corresponding production files should be loaded into the browser and in which order.
5. Start the test server and capture a browser.
6. Launch unit tests and monitor test results in the Run tool window.

The topics in this part provide guidelines in JavaScript-specific unit testing procedures.

- Enabling JavaScript Unit Testing Support
- Creating JavaScript Unit Tests
- Running JavaScript Unit Tests
- Monitoring Code Coverage for JavaScript
- Debugging JavaScript Unit Tests

For general information on testing in PhpStorm, see the section Testing .

Enabling JavaScript Unit Testing Support

With PhpStorm, you can run and debug unit tests using one of the following test runners:

- JSTestDriver
- Karma
- Cucumber.js

To write unit tests that are recognized and processed by these test runners, use the following JavaScript unit testing frameworks:

- JSTestDriver Assertion framework. The framework libraries are bundled with the JSTestDriver plugin.
- Jasmine .
- QUnit .
- Mocha

In this part:

- Preparing to Use Cucumber.Js Test Runner
- Preparing to Use JSTestDriver Test Runner
- Preparing to Use Karma Test Runner
- Preparing to Use Mocha Test Framework

Preparing to Use Cucumber.js Test Runner

The Cucumber.js test runner supports executing unit tests against the Node.js server. Cucumber.js runs tests that are called features and are written in the Gherkin language. Each feature is described in a separate file with the extension `feature`. Feature files are marked with the icon.

The easiest way to install the Cucumber.js test runner is to use the Node Package Manager (npm), which is a part of Node.js. See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Cucumber.js test runner executable file, choose one of the following methods:

- Install the test runner globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the test runner in a specific project and thus restrict its use to this project.
- Install the test runner in a project as a development dependency.

In either installation mode, make sure that the parent folder of the Cucumber.js test runner is added to the `PATH` variable. This enables you to launch the test runner from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

On this page:

- Preparing to install Cucumber.js
- Installing Cucumber.js globally
- Installing Cucumber.js in a project
- Installing Cucumber.js as a development dependency

Preparing to install Cucumber.js

1. Download and install Node.js. The runtime environment is required for two reasons:
   - The Cucumber.js test runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Cucumber.js test runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Cucumber.js test runner and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Install and enable the Cucumber.js and Gherkin plugins. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Installing Cucumber.js globally

Global installation makes the test runner available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the test runner is automatically added to the `PATH` variable, which enables you to launch the test runner from any folder. To install the test runner globally, do one of the following:

Run the installation from the command line in the global mode:

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
3. Type the following command at the command line prompt:

   ```
   npm install -g cucumber
   ```

   The `-g` key makes the test runner run in the global mode. Because the installation is performed through NPM, the Cucumber.js test runner is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the test runner from any folder.

   For more details on the NPM operation modes, see npm documentation . For more information about installing the Cucumber.js test runner, see https://npmjs.org/package/cucumber .

Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `cucumber` package.
4. Select the Options check box and type `-g` in the text box next to it.
5. Optionally specify the product version and click Install Package to start installation.

Installing Cucumber.js in a project

Installing the test runner in a specific project restricts its use to this project. To run project installation, do one of the following:

Run the installation from the command line:

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
3. Type the following command at the command line prompt:

```
npm install cucumber
```

Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `cucumber` package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the Cucumber.js test runner in a `node_modules` folder. If your project already contains such folder, the Cucumber.js test runner is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the Cucumber.js test runner in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the Cucumber.js test runner is installed there.

In either case, make sure that the parent folder of the Cucumber.js test runner is added to the `PATH` variable. This enables you to launch the test runner from any folder.

Installing Cucumber.js as a development dependency

Because Cucumber.js is a test framework, which is of no need for those who are going to re-use your application, it is helpful to have it excluded from download for the future. This is done by marking the tool as a development dependency , which actually means adding the tool in the `devDependencies` section of the `package.json` file.

With PhpStorm, you can have the test runner marked as a development dependency right during installation. Do one of the following:

– Run the installation from the command line in the global mode:

1. Launch the embedded Terminal by hovering your mouse pointer over ▣ in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
3. Type the following command at the command line prompt:

```
npm install --dev cucumber
```

– Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `cucumber` package.
4. Select the Options check box and type `--dev` in the text box next to it.
5. Optionally specify the product version and click Install Package to start installation.

After installation, the test runner is added to the `devDependencies` section of the `package.json` file.

Preparing to Use JSTestDriver Test Runner

On this page:

- Configuring the JSTestDriver test runner
- Configuring testing frameworks in a project

Configuring the JSTestDriver test runner

You do not need to download the JSTestDriver framework manually. The server and the assertion framework are provided through the JSTestDriver plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

This plugin does the following:

- Runs the JSTestDriver server that captures an opened browser to execute tests in.
- During the test creation, detects the unit testing framework the test code complies with, whereupon suggests the Add <test framework> support intention action , provided that the framework is recognized as a PhpStorm JavaScript library and is thus available in the IDE.

Configuring testing frameworks in a project

1. Download the framework of your choice and configure it as a PhpStorm JavaScript library .
2. Do one of the following:
   - Add the project folder to the library scope .
   - Enable the framework support on-the-fly during test creation . Write the test as required, position the cursor at it, and press `Alt+Enter`. Then choose the Add <test framework> support intention action from the list.
   - To use Jasmine, add `jasmine-jstd-adapter` to the configuration file.
     Open `jsTestDriver.conf` and type the following code in it:

```
load:

lib/jasmine/jasmine.js
lib/jasmine-jstd-adapter/JasmineAdapter.js
```

For more details about JSTestDriver configuration files, see Running JavaScript Unit Tests .

Preparing to Use Karma Test Runner

The Karma test runner supports executing unit tests against the Node.js server. The tests themselves can use Jasmine , QUnit , or Mocha libraries in Karma-specific edition.

Besides running unit tests, with Karma you can measure how much of your code is covered with tests. For more details, see Monitoring Code Coverage for JavaScript . To run tests with coverage, you need an additional Karma-related package.

The easiest way to install the Karma test runner is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Karma test runner executable file, choose one of the following methods:

- Install the test runner globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the test runner in a specific project and thus restrict its use to this project.
- Install the test runner in a project as a development dependency .

In either installation mode, make sure that the parent folder of the Karma test runner is added to the `PATH` variable. This enables you to launch the test runner from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

On this page:

- Preparing to install the Karma test runner
- Installing Karma globally
- Installing Karma in a project
- Configuring testing frameworks in a project

Preparing to install the Karma test runner

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Karma test runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Karma test runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Karma test runner and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Install and enable the Karma repository plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing Karma globally

Global installation makes a test runner available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the test runner is automatically added to the `PATH` variable, which enables you to launch the test runner from any folder. To install the test runner globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g karma
     ```

     The `-g` key makes the test runner run in the global mode. Because the installation is performed through NPM, the Karma test runner is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the test runner from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the Karma test runner, see https://npmjs.org/package/karma .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing Karma in a project

Installing a test runner in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install karma
```

– Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the required package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the Karma test runner in a `node_modules` folder. If your project already contains such folder, the Karma test runner is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the Karma test runner in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the Karma test runner is installed there.

In either case, make sure that the parent folder of the Karma test runner is added to the `PATH` variable. This enables you to launch the test runner from any folder.

Configuring testing frameworks in a project

1. Download the Jasmine, QUnit, or Mocha framework. The easiest way is to use the Node.js Package Manager:
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
   2. On the Node.js and NPM page that opens, click Install in the Packages area.
   3. In the Available Packages dialog box that opens, select `karma-jasmine`, `karma-qunit`, or `karma-mocha` package and click Install Package. Close the dialog box when ready. PhpStorm returns to the Node.js page, where the selected package is added to the Packages list. Click OK.
2. To enable PhpStorm to resolve references to the downloaded framework and provide code completion and other types of coding assistance, configure the framework as a PhpStorm JavaScript library .

Preparing to Use Mocha Test Framework

The Mocha test framework supports executing unit tests against the Node.js server. The tests themselves can use BDD, TDD, Exports, and QUnit interfaces.

The easiest way to install the Mocha test framework is to use the Node Package Manager (npm), which is a part of Node.js. See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Mocha test framework executable file, choose one of the following methods:

- Install the test framework globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the test framework in a specific project and thus restrict its use to this project.
- Install the test framework in a project as a development dependency.

In either installation mode, make sure that the parent folder of the Mocha test framework is added to the `PATH` variable. This enables you to launch the test framework from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

On this page:

- Preparing to install the Mocha test framework
- Installing Mocha globally
- Installing Mocha in a project

Preparing to install the Mocha test framework

1. Download and install Node.js. The runtime environment is required for two reasons:
   - The Mocha test framework is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Mocha test framework.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Mocha test framework and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins.

Installing Mocha globally

Global installation makes a test framework available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the test framework is automatically added to the `PATH` variable, which enables you to launch the test framework from any folder. To install the test framework globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs.
  3. Type the following command at the command line prompt:

     ```
     npm install -g mocha
     ```

     The `-g` key makes the test framework run in the global mode. Because the installation is performed through NPM, the Mocha test framework is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the test framework from any folder.

     For more details on the NPM operation modes, see npm documentation. For more information about installing the Mocha test framework, see https://npmjs.org/package/mocha.
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing Mocha in a project

Installing a test framework in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

     ```
     npm install mocha
     ```
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.

2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the required package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the Mocha test framework in a `node_modules` folder. If your project already contains such folder, the Mocha test framework is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the Mocha test framework in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the Mocha test framework is installed there.

In either case, make sure that the parent folder of the Mocha test framework is added to the `PATH` variable. This enables you to launch the test framework from any folder.

Creating JavaScript Unit Tests

PhpStorm provides integration with JavaScript unit testing frameworks thus providing rich coding assistance for writing JavaScript unit tests. To get framework-aware coding assistance, download the desired framework, configure it as a PhpStorm JavaScript library, and define its visibility and scope.

The visibility status of a library determines whether it can be used in one project (Project) or can be re-used at the IDE level (Global).

The scope of a library defines the set of files and folders in which the library is considered as library, that is, it is write-protected, excluded from check for errors and refactoring, but only affects the completion list and highlighting.

For details, see Configuring JavaScript Libraries .

To create JavaScript unit tests, perform these general steps

- Create a folder `test` at the same level as the `src` folder.
- Populate the `test` folder. For each production file, create a separate test file and name it as follows: `<name of production file>.<Test>.js`.
- Mark the folder where the tests are stored as test folder .
- Mark the folder where the tests are stored as a test source folder, see Configuring Content Roots .
- If necessary, have PhpStorm detect and enable the required testing framework on-the-fly .

Running JavaScript Unit Tests

JavaScript unit tests are run in a browser against a test server which actually handles the testing process.

This server allows you to capture a browser to run tests in, loads the test targets to the captured browser, controls the testing process, and exchanges data between the browser and PhpStorm, so you can view test results without leaving the IDE.

The test server does not necessarily have to be on your machine, it can be launched right from PhpStorm through a test runner. Currently, PhpStorm supports integration with two test runners: JSTestDriver and Karma .

In this part:

- Running Cucumber.Js Unit Tests
- Running Unit Test on JSTestDriver
- Running Unit Tests on Karma
- Running Mocha Unit Tests

Running Cucumber.js Unit Tests

JavaScript unit tests are run in a browser against a test server which actually handles the testing process.

This server allows you to capture a browser to run tests in, loads the test targets to the captured browser, controls the testing process, and exchanges data between the browser and PhpStorm, so you can view test results without leaving the IDE.

The test server does not necessarily have to be on your machine, it can be launched right from PhpStorm through the Cucumber.js test runner. Cucumber.js runs tests that are called features and are written in the Gherkin language. Each feature is described in a separate file with the extension `feature`.

On this page:

- Before you start
- Creating a Cucumber.js run configuration
- Launching unit tests

Before you start

1. Download and install Node.js. The runtime environment is required for two reasons:
   - The Cucumber.js test runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Cucumber.js test runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Cucumber.js test runner and npm from any folder.

2. Make sure the Cucumber.js test runner is installed as described in Preparing to Use Cucumber.Js Test Runner.

3. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins.

4. Install and enable the Cucumber.js and Gherkin plugins. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins. Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Creating a Cucumber.js run configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Open the test file in the editor, and then choose Create <file name> on the context menu.
   - Select the test file in the Project tool window, and then choose Create <file name> on the context menu of the selection.

2. Click the Add button + on the toolbar and select the Cucumber.js configuration type. The Run/Debug Configuration: Cucumber.Js dialog box opens.

3. In the Feature file or directory text box, specify the tests to run. Cucumber.js runs tests that are called features and are written in the Gherkin language. Each feature is described in a separate file with the extension `feature`.
   - To have one feature executed, specify the path to the corresponding `.feature` file.
   - To have a bunch of features executed, specify the folder where the `.feature` files to run are stored.

4. In the Executable path text box, specify the location of the cucumber executable file, `.cmd`, `.bat`, or other depending on the operating system used. The location depends on the installation mode, see Installing the Cucumber.js test runner.

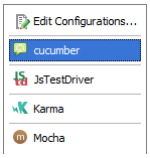
5. Optionally, specify the command line arguments to be passed to the Cucumber.js executable file, such as `-r` (`--require LIBRARY|DIR`), `-t` (`--tags TAG_EXPRESSION`), or `--coffee`. For details, see Cucumber's native built-in help available through the following command:

```
cucumber-js --help
```

6. Apply the changes and close the dialog box.

Launching unit tests

1. To launch the tests according to a run configuration, select the Cucumber.js run/debug configuration from the list on the main toolbar.



   Then click the Run button ▶ to the right of the list.

2. The test server starts automatically without any steps from your side. View and analyze messages from the test server in the Run tool window.

3. Monitor test execution in the Test Runner tab of the Run tool window as described in Monitoring and Managing Tests.

Running Unit Test on JSTestDriver

JavaScript unit tests are run in a browser against a test server which actually handles the testing process.

This server allows you to capture a browser to run tests in, loads the test targets to the captured browser, controls the testing process, and exchanges data between the browser and PhpStorm, so you can view test results without leaving the IDE.

The test server does not necessarily have to be on your machine, it can be launched right from PhpStorm through the JSTestDriver test runner.

The test server loads tests to the browser according to the configuration files that define which test and corresponding production files should be loaded and in which order. JSTestDriver server treats `*.jstd` and `*.conf` files as test runner configuration files. Create one or several configuration files in advance manually and then specify them in the run configuration .

On this page:

- Before you start
- Creating a test runner configuration file manually
- Starting a PhpStorm default JSTestDriver test server
- Capture a browser to execute tests in
- Creating a JSTestDriver run configuration
- Launching unit tests

Before you start

- Make sure the JSTestDriver repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Creating a test runner configuration file manually

A test runner configuration file defines which test and corresponding production files should be loaded and in which order. Any file with the extension `*.jstd` or `*.conf` is treated as a test runner configuration file.

1. In the Project tree, select the parent folder of the `src` (production) and the `test` folders, and choose New | File on the context menu.
2. In the New File dialog box, that opens, type a name of the configuration file with the extension `jstd` or `conf`.
3. Open the new file in the editor and specify the full path to the current folder and paths to the files to load relative to the current folder. Use wildcards to specify file name patterns. The required format is YAML , for more details, see description of test runner configuration files .

Starting a PhpStorm default JSTestDriver test server

If you are going to use a test server running on another machine or listening to another port, start it according to the server-specific instructions and specify its URL address in the Server ares of the Run/Debug Configuration: JSTestDriver dialog box.

1. Make sure the JSTestDriver plugin is enabled and at least one test runner configuration file is available in the project.
2. Open the JSTestDriver Server tool window by doing one of the following:
   - On the main menu, choose View | Tool Windows | JSTestDriver Server.
   - Click the Run button on the toolbar, and then click the Start a local server link in the pop-up window that opens.
3. In the JSTestDriver Server tool window, that opens, click the Run a local server toolbar button.

To stop the server when you are through with unit testing, click the Stop the local server toolbar button.

Capture a browser to execute tests in

1. Start the JSTestDriver Server if it is not running yet, and then switch to the JSTestDriver Server tool window.
2. To start a local browser with the Remote Console of the JSTestDriver, do one of the following:
   - Click the icon that indicates the browser of your choice.
   - If the browser is already opened, copy the contents of the Capture a browser using the URL read-only field and paste the URL in the address bar.

   In either case, the icon that indicates the chosen browser becomes active.

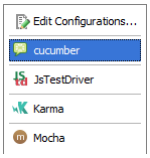
   You can have several browsers captured. However if you are going to debug your unit tests, you will have to appoint a browser for debugging in the Debug tab of the Run/Debug Configuration: JSTestDriver dialog box.
3. Switch to the JSTestDriver Server tool window and click the icon that indicates the browser you just opened. PhpStorm displays a message informing you that it is ready for executing tests.

Creating a JSTestDriver run configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Open the test file in the editor, and then choose Create <file name> on the context menu.
   - Select the test file in the Project tool window, and then choose Create <file name> on the context menu of the selection.
2. Click the Add button on the toolbar and select the JSTestDriver configuration type.
3. In the dialog box that opens, specify the test scope, configuration parameters, and activities to perform before test execution.
4. Apply the changes and close the dialog box.

Launching unit tests

1. To launch the tests according to a run configuration, select the JSTestDriver run/debug configuration from the list on the main toolbar.



   Then click the Run button to the right of the list.

2. To launch a single test, open the test file in the editor and click the blue arrow icon in the gutter area next to the text to run.

   Note The arrows appear only if the test framework used in your tests is associated with the project, so PhpStorm can recognize the tests, see Enabling JavaScript Unit Testing Support for details.

3. Monitor test execution in the Test Runner tab of the Run tool window as described in Monitoring and Managing Tests .

Running Unit Tests on Karma

JavaScript unit tests are run in a browser against a test server which actually handles the testing process.

This server allows you to capture a browser to run tests in, loads the test targets to the captured browser, controls the testing process, and exchanges data between the browser and PhpStorm, so you can view test results without leaving the IDE.

The test server does not necessarily have to be on your machine, it can be launched right from PhpStorm through the Karma test runner.

Karma executes unit tests according to a `karma.conf.js` configuration file which is generated semi-automatically in the interactive mode.

On this page:

- Before you start
- Generating a Karma configuration file
- Creating a Karma run configuration
- Launching unit tests

Before you start

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Karma test runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Karma test runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Karma test runner and npm from any folder.

2. Make sure the Karma test runner is installed as described in Preparing to Use Karma Test Runner .

3. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

4. Install and enable the Karma repository plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Generating a Karma configuration file

Karma executes unit tests according to a `karma.conf.js` configuration file which is generated semi-automatically in the interactive mode. The instruction below ensures successful creation of a consistent configuration file `karma.conf.js` which in its turn ensures successful execution of the tests in your project. For more details, see http://karma-runner.github.io/0.10/config/configuration-file.html .

1. In the command line mode, switch to your project directory.

2. Type the following command at the command line prompt:

```
karma init
```

   If Karma does not start, check the installation: the parent folder or the Karma executable file should be specified in the `PATH` variable.

3. Answer the questions to specify the following basic settings:
   - The testing framework to use.
   - The browsers to be captured automatically.
   - The patterns that define the location of test files to be involved in testing or excluded from it, for example, `src/*.js` and `test/*.js` . For more details, see http://karma-runner.github.io/0.10/config/files.html .

Creating a Karma run configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Open the test file in the editor, and then choose Create <file name> on the context menu.
   - Select the test file in the Project tool window, and then choose Create <file name> on the context menu of the selection.

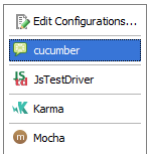
2. Click the Add button + on the toolbar and select the Karma configuration type.

3. In the dialog box that opens, specify the location of the Node.js and Karma executable files and the path to the `karma.conf.js` configuration file.

4. Apply the changes and close the dialog box.

Launching unit tests

1. To launch the tests according to a run configuration, select the Karma run/debug configuration from the list on the main toolbar.



   Then click the Run button ▶ to the right of the list.

2. The Karma test server starts automatically without any steps from your side. View and analyze messages from the test server in the Karma Server tab of the

Run tool window.

3. Monitor test execution in the Test Runner tab of the Run tool window as described in Monitoring and Managing Tests .

Running Mocha Unit Tests

On this page:

- Before you start
- Creating a Mocha run configuration

Before you start

1. Make sure the Node.js runtime environment and the Mocha framework are installed on your computer.
2. Make sure the NodeJS repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

See Preparing to Use Mocha Test Framework for details.

Creating a Mocha run configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose Run | Edit Configurations.
   - Open the test file in the editor, and then choose Create <file name> on the context menu.
   - Select the test file in the Project tool window, and then choose Create <file name> on the context menu of the selection.

   Click the Add button + on the toolbar and select the Mocha configuration type. The Run/Debug Configuration: Mocha dialog box opens.
2. Specify the location of the Node.js executable file and the Mocha package.
3. Specify the working directory of the application. All references in test scripts will be resolved relative to this folder, unless such references use full paths.
   By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
4. Specify the tests to run:
   - All in directory: choose this option to run all the tests from files stored in a folder. In the Test directory field, specify the folder with the tests. To have PhpStorm look for tests in the subfolders under the specified directory, select the Include subdirectories check box.
   - File pattern: choose this option to have PhpStorm look for tests in all the files with the names that match a certain mask and specify the mask in the Test file pattern field.
   - Test file: choose this option to run only the tests from one file and specify the path to this file in the Test file field.
   - Suite: choose this option to run individual suites from a test file. In the Test file field, specify the file where the required suites are defined. Click the Suite name field and configure a list of suites to run. To add a suite to the list, click + and type the name of the required suite. To remove a suite, select it in the list and click –
   - Test: choose this option to run individual tests from a test file. In the Test file field, specify the file where the required tests are defined. Click the Test name field and configure a list of tests to run. To add a test to the list, click + and type the name of the required test. To remove a test, select it in the list and click –
5. Choose the interface used in the test to run.
6. Apply the changes and close the dialog box.

Monitoring Code Coverage for JavaScript

Code Coverage shows how much of your code is covered with tests and marks covered and uncovered lines visually right in the editor.

In this section:

- Prerequisites
- Enabling running tests with coverage in Karma
- Measuring code coverage for tests executed on JSTestDriver
- Measuring code coverage for tests executed on Karma

Prerequisites

1. The JSTestDriver or Karma test runner is installed and configured.
2. Additional third-party testing frameworks, for example, Jasmine , QUnit , or Mocha , are downloaded and integrated with PhpStorm .
3. A runner-specific configuration file is generated or written manually.

For more details, see Preparing to Use Karma Test Runner or Preparing to Use JSTestDriver Test Runner depending on your choice.

Enabling running tests with coverage in Karma

Coverage for tests run in Karma is calculated by the istanbul coverage tool. The tool is shipped within the `karma-coverage` package. Therefore running tests with coverage in Karma requires that you download the `karma-coverage` package and manually update the `karma.conf.js` configuration file.

The `karma-coverage` package should be installed in the same folder with Karma (`karma (library home)` package), no matter whether Karma is installed globally or in your project. Keep this restriction in mind when installing `karma-coverage` manually in the command line mode. If you are installing `karma-coverage` through the Node Package Manager interface, the proper installation folder will be chosen automatically.

1. Install the `karma-coverage` package through the Node Package Manager (npm).
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
   2. On the Node.js and NPM page that opens, click Install in the Packages area.
   3. In the Available Packages dialog box that opens, select the `karma-coverage` package, click Install Package, and close the dialog box when ready. PhpStorm returns to the Node.js page, where the `karma-coverage` package is added to the Packages list. Click OK.

   For details, see Installing and Removing External Software Using Node Package Manager .

2. Open the `karma.conf.js` configuration file and add the following information to it manually:
   1. Locate the `reporters` definition and add `coverage` to the list of values in the following format:

   ```
   reporters: ['progress', 'coverage'],
   ```

   2. Add a `preprocessors` definition and specify the coverage scope according to the following format:

   ```
   preprocessors: {
                       '**/*.js':
                       ['coverage']
                       }
   ```

Measuring code coverage for tests executed on JSTestDriver

1. Prepare tests manually or have tests generated by PhpStorm.
2. Create a JSTestDriver configuration file `*.jstd` or `*.conf` . For details, see JSTestDriver Configuration file .
3. Create a run configuration of the type JSTestDriver .
   To have specific files excluded from coverage analysis, specify the paths to them in the Coverage tab of the Run/Debug Configuration: JSTestDriver dialog box.
4. Start the JSTestDriver server and capture a browser to run the tests in.
5. On the main toolbar, select the JSTestDriver run configuration in the Run/Debug Configurations drop-down list and click the Run with Coverage button .
6. Monitor the code coverage in the Coverage tool window.

Measuring code coverage for tests executed on Karma

1. Prepare tests manually or have tests generated by PhpStorm.
2. Create a Karma configuration file `karma.conf` . For details, see Running Unit Tests on Karma .
3. Create a run configuration of the type Karma .
4. On the main toolbar, select the Karma run configuration in the Run/Debug Configurations drop-down list and click the Run with Coverage button .
5. Monitor the code coverage in the Coverage tool window. Note that when the tests are executed on `Karma` the Coverage tool window does not contain the Generate Coverage Report toolbar button because a coverage report is actually generated on the disk every time Karma tests are run. The format of a coverage report can be configured in the configuration file, for example:

```
// karma.conf.js
                    module.exports = function(config) {

                    config.set({ ...

                    // optionally, configure the reporter

                    coverageReporter: { type : 'html', dir
                    : 'coverage/'
                    }

                    ...

                    });};
```

The following `type` values are acceptable:

- `html` produces a bunch of HTML files with annotated source code
- `lcovonly` produces an `lcov.info` file
- `lcov` produces HTML + `.lcov` files. This format is applied by default.
- `cobertura` produces a `cobertura-coverage.xml` file for easy Hudson integration
- `text-summary` produces a compact text summary of coverage, typically to the console
- `text` produces a detailed text table with coverage for all files

Debugging JavaScript Unit Tests

Besides simply executing unit tests on the JSTestDriver or Karma without the ability to analyze the problems that arise, PhpStorm provides the ability to debug unit tests just like JavaScript source code.

On this page:

– Prerequisites

– Debugging unit tests executed on JSTestDriver

– Debugging unit tests executed on Karma

Prerequisites

1. The JSTestDriver or Karma test runner is installed and configured.
2. Additional third-party testing frameworks, for example, Jasmine , QUnit , or Mocha , are downloaded and integrated with PhpStorm .
3. A runner-specific configuration file is generated or written manually.

For more details, see Preparing to Use Karma Test Runner or Preparing to Use JSTestDriver Test Runner depending on your choice.

Debugging unit tests executed on JSTestDriver

1. Prepare the unit tests to run.
2. In the unit tests, set the breakpoints for PhpStorm to postpone test execution at.
3. Create a JSTestDriver configuration file `*.jstd` or `*.conf` . For details, see JSTestDriver Configuration file .
4. Start a PhpStorm default JSTestDriver test server .
5. Capture a browser to execute the tests in .
6. Create a JSTestDriver run configuration . If you have captured two browsers, specify the browser for debugging the tests: in the Debug tab of the Run/Debug Configuration: JSTestDriver dialog box, choose the relevant browser from the Debug drop-down list. The available options are Chrome and Dartium.
7. Choose Run | Debug on the main menu, or click Debug button on the toolbar, or press `Shift+F9` .
8. In the Debug tool window that opens, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , etc.

Debugging unit tests executed on Karma

1. Prepare tests manually or have tests generated by PhpStorm.
2. Create a Karma configuration file `karma.conf` . For details, see Running Unit Tests on Karma .
3. Create a run configuration of the type Karma .
4. On the main toolbar, select the Karma run configuration in the Run/Debug Configurations drop-down list and click Debug button on the toolbar, or press `Shift+F9` .
5. In the Debug tool window that opens, proceed as usual: step through the program , stop and resume program execution, examine it when suspended , etc.

Minifying JavaScript

The term minification or compression in the context of JavaScript means removing all unnecessary characters, such as spaces, new lines, comments without affecting the functionality of the source code.

These characters facilitate working with the code at the development and debugging stage by improving the code readability. However at the production stage these characters become extraneous: being insignificant for code execution, they increase the size of code to be transferred. Therefore it is considered good practice to remove them before deployment.

PhpStorm supports integration with the following JavaScript minification tools:

- YUI Compressor
- UglifyJS
- Closure Compiler

In PhpStorm, minifier configurations are called File Watchers. For each supported minifier, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a minifier against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the minifier to use on your machine.

On this page:

- Installing and configuring the Closure Compiler minification tool
- Installing and configuring the YUI Compressor JS or UglifyJS minification tool
- Creating a file watcher
- Minifying the code

Installing and configuring the Closure Compiler minification tool

1. Download and install the Node.js runtime environment because the JavaScript minifier is started through Node.js.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.
4. Download the compiled tool at http://dl.google.com/closure-compiler/compiler-latest.zip .
5. Extract the archive to the folder under the Node.js. This ensures that Node.js, which is required for starting the tool, will be specified in the path to it.

Installing and configuring the YUI Compressor JS or UglifyJS minification tool

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The JavaScript minifier is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the JavaScript minifier.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the JavaScript minifier and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.
4. Download and install the JavaScript minification tool. The easiest way is to use the Node Package Manager (npm), which is a part of Node.js .
   1. Switch to the directory where the Node Package Manager (npm) is stored or define a `path` variable for it so it is available from any folder.
   2. Depending on the tool you want to use, type one of the following commands at the command line prompt:
      - To have the YUI Compressor JS installed, type:

      ```
      npm install yuicompressor
      ```

      - To have the UglifyJS installed, type:

      ```
      npm install uglify-js
      ```

   If you use the Node Package Manager (npm), the tools are installed under Node.js so Node.js, which is required for starting the tool, will be specified in the path to it.

Creating a file watcher

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
2. Click the Add button + or press `Alt+Insert`. Depending on the tool you are going to use, choose the appropriate predefined template from the pop-up list:
   - Closure Compiler
   - UglifyJS
   - YUI Compressor JS
3. In the Program text box, specify the path to the compiler executable file or archive depending on the chosen predefined template:
   - `compiler.jar` for Closure Compiler

- `yuicompressor-<version>.jar` for YUI Compressor JS
- `uglifyjs.cmd` for UglifyJS

If you installed the tool through the Node Package Manager, PhpStorm locates the required file itself and fills in the field automatically. Otherwise, type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.

4. Proceed as described on page Using File Watchers .

Minifying the code

When a Minification File Watcher is enabled (see Enabling and disabling File Watchers), minification starts automatically as soon as a file to which transpilation is applicable is changed or saved, see Configuring the behaviour of the File Watcher .

PhpStorm creates a separate file with the generated output. The file has the name of the source JavaScript file and the extension `min.js` . The location of the generated file is defined in the Output paths to refresh text box of the New Watcher dialog . However, in the Project Tree, it is shown under the source JavaScript file which is now displayed as a node.

Tracing with Spy-js

Spy-js is a tool for JavaScript developers that allows to simply debug/trace/profile JavaScript running on different platforms/browsers/devices. The tool fills gaps that existing browser development tools have and tackles common development tasks from a different angle.

The tool also traces NodeJS, server-side, applications.

On this page:

- Preparing for tracing with Spy-js
- Spy-js basics
- Spy-js UI
- Spy-js tracing session
  - Initiating a Spy-js tracing session
    - Launching a tracing session of a Web application
    - Launching a tracing session of a NodeJS application
  - Saving and loading tracing sessions
    - To save an image of a tracing session
    - To load an image of a previous tracing session
  - Configuring the range of events to display
    - Defining a new event filter
    - Choosing a filter to apply
    - Creating exclusion rules on the fly
    - Setting timestamp labels
  - Synchronization and navigation between the oanes and the editor
    - Navigating from an event or a script to the trace file
    - Navigating from an event or script to the source file
    - Navigating from a function to its call
    - Navigating through ECMAScript 6, TypeScript, or CoffeeScript
    - Advanced trace navigation
    - Advanced trace search
  - Expanding the basic completion list with runtime data (Spy-js autocompletion)
    - Activating Spy-js autocompletion
  - Evaluating expressions without running a debugging session (Spy-js magnification)
    - Activating Spy-js magnification
- Viewing dependency diagrams

Preparing for tracing with Spy-js

1. Download and install NodeJS because it is used by the Spy-js trace server.
2. Make sure the Spy-js repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Spy-js basics

With Spy-js, you can trace both Web applications and NodeJS applications. Although the Spy-js UI in either case is the same, the mechanisms of the tracing differ.

In order to trace a script, Spy-js has to modify it on the fly. The modification does not affect the logic of the script, it is just inserting instrumentation instructions that report back to Spy-js UI about what functions have been invoked when the script executes.

- To modify website scripts, Spy-js has to act as a proxy server that "sits" between your browser and the website you are tracing. When you open a traced website in your browser, Spy-js receives the script request, requests the script from your website, receives the script, makes the required modifications, and sends it back to your browser where the script executes, and sends the runtime information to the Spy-js UI.
  The proxy server can be configured automatically by selecting the Automatically configure system proxy check box in the Run/Debug Configuration: Spy-Js dialog or manually. See how to configure proxy settings manually on Windows , Mac , Ubuntu , iOS , Android , Windows Phone . Please note that some desktop browsers have their own screens for proxy settings configuration.
- In case of a NodeJS application, Spy-js cannot get between the NodeJS server and the scripts if the application is already running. Therefore to trace a NodeJS application, spy-js launches the NodeJS server and the application itself. This enables Spy-js to intercept and modify script requests and scripts, whereupon the tracing procedure runs as when tracing a website script.

You can also trace applications with ECMASript6, CoffeeScript, and TypeScript code: Spy-js recognizes source maps that set correspondence between the original code and the JavaScript code generated during transpilation, see Using File Watchers for details.

A Spy-js session is initiated from PhpStorm through a run configuration, see Initiating a Spy-js Session .

Spy-js UI

All the tracing-related activities, such as viewing captured events, examining their call stacks, navigating to the source code, etc. are performed in the dedicated Spy-Js Tool Window , in particular in its Trace Run Tab . The tab consists of a toolbar and three panes:

- Events Pane
  The pane shows a tree of captured events. The top-level nodes represent documents that are Web pages involved in tracing. When you hover the mouse over a document, PhpStorm displays a tooltip with the URL address of the document, the browser in which it is opened, and the operating system the browser is running on. The document node is also supplied with an icon that indicates the browser in which it is opened.

  Under each document node, events detected on the page and scripts started from it are listed. Events of the same type are grouped into visual containers. The header of a container displays the name of the events grouped in it, the average execution time across all the events within the container, and the number of events inside the

container. You can expand each node and inspect the individual events in it.

Script file names have different colour indicators to help distinguishing between them when working with the Event Stack pane. By hovering your mouse over a script file name, you can see the full script URL.

Once an event is clicked, its call stack is displayed in the Event Stack pane. The stack is represented by a tree of function calls.

- Event Stack Pane

  Once an event in the Events pane is clicked, its call stack is displayed in the Event Stack pane. The stack is represented by a tree of function calls. Each tree node represents the invoked function. Node text contains the total execution time, the script file name and the function name. When you click a node, the Quick Evaluation pane shows additional function call details, parameter values and return value, occurred exception details if there was one during the function execution.

  The pane is synchronized with the editor, so you can navigate from an item in the stack tree to the corresponding trace file or source file.

  - A trace file is a write-protected prettified version of the script selected in the Events pane or the script whose function is double clicked in the Event Stack pane. A trace file is named `<file name>.js.trace`. When you double click an item in the stack tree or select it and choose Jump to Trace on the context menu of the selection, the corresponding trace file opens in the editor with the cursor positioned at the clicked function. Another approach is to press the Autoscroll to Trace toggle button and select various stack nodes. In this case, the trace file opens when you click an event or script in the Events pane.
    You can not only jump to a function but also to the place in the code where it was called from. To do that, select the required item and choose Jump to Caller on the context menu.

    The contents of the file are highlighted to display the code execution path of the selected stack node.
  - When you are tracing an application with ECMASript6, CoffeeScript, and TypeScript code, Spy-js also generates mapped trace files. These are EcmaScript 6, TypeScript, or CoffeeScript trace files with the extensions `.ts.trace`, `.coffee.trace`, or `.js.trace`. The fragments of code in these files are highlighted as if they were really executed.
  - You can also navigate to the source file displayed as is, without prettifying, by selecting an item in the Event Stack pane and choosing Jump to Source on the context menu of the selection. If the traced site is mapped with a PhpStorm project, PhpStorm detects the corresponding local file according to the mapping and opens this file in the editor. If you are tracing a site that is not mapped to a PhpStorm project, PhpStorm opens the read-only page source, just as if you chose View Page Source in the browser.
    When the traced site is mapped with a PhpStorm project, PhpStorm opens the source file on any attempt to edit the opened trace file.

- Quick Evaluation Pane

  When you click a node in the Event Stack pane, the Quick Evaluation pane shows additional function call details, parameter values and return value, occurred exception details if there was one during the function execution.

Spy-js tracing session

Initiating a Spy-js tracing session

The way you initiate a tracing session depends on the type of the application you are going to trace.

Launching a tracing session of a Web application

To a trace an application with CoffeeScript or TypeScript code, you need to transpile the original code into JavaScript, see and Transpiling TypeScript to JavaScript for details.

1. Create a run configuration of the type Spy-js. To do that, choose Run | Edit Configurations on the main menu, click the Add New Configuration toolbar button +, and choose Spy-js on the context menu. In the Run/Debug Configuration: Spy-Js dialog box that opens, specify the following:
   1. The location of the Node interpreter.
   2. The trace server port. This port number must be the same as your system proxy port. If the Automatically configure system proxy check box is selected, the specified port number is automatically set for the system proxy server. Otherwise you will have to specify the value of the field in the system proxy settings manually.
      The trace server port is filled in automatically. To avoid port conflicts, it is recommended that you accept the suggested value and keep the Automatically configure system proxy check box selected.
   3. The way to configure the proxy server.
      To have the system proxy server activated automatically with the port specified in the Trace server port field, select the Automatically configure system proxy check box.

      Clear the Automatically configure system proxy check box to specify proxy settings manually. See how to configure proxy settings manually on Windows, Mac, Ubuntu, iOS, Android, Windows Phone. Please note that some desktop browsers have their own screens for proxy settings configuration.
   4. From the Use dropdown list, choose the way to specify the way to configure a tracing session.
      - To have Spy-js apply its internal predefined configuration, choose Default configuration.
      - To have your custom manually created configuration applied, choose the Configuration file option and then specify the location of your custom configuration file in the Configuration field below.
        A configuration file is a JavaScript file with the extension `.js` or `.conf.js` that contains valid JavaScript code that meets the Spy-js configuration requirements. If PhpStorm detects files with the extension `.conf.js` in the project, these files are displayed in the drop-down list.

        Type the path to the configuration file manually or click the Browse button ... and choose the location in the dialog box that opens. Once specified, a configuration file is added to the drop-down list so you can get if next time from the list instead of specifying the path.

2. To launch a tracing session, click the Run toolbar button ▶. The Spy-Js Tool Window opens with an empty Trace Run tab and a Trace Proxy Server tab informing you about the status of the proxy server.

3. Switch to the browser and refresh the page to start debugging from. Spy-js starts capturing events on this page and the Spy-js tool window shows them in the Events pane.

Launching a tracing session of a NodeJS application

1. Create a run configuration of the type Spy-js for NodeJS. To do that, choose Run | Edit Configurations on the main menu, click the Add New Configuration toolbar button +, and choose Spy-js for NodeJS on the context menu. In the Run/Debug Configuration: Spy-js for NodeJS dialog box that opens, specify the following:

1. The location of the Node interpreter.
2. The Node parameters, that is, the NodeJS-specific command line options to be passed to the NodeJS executable file. For example, to enable tracing ECMAScript 6 scripts, specify `--harmony` as a Node parameter. Note that Node.js must be version 0.11.13 or higher.
3. The working directory of the application. All references in the starting file file, for example, imports, will be resolved relative to this folder, unless such references use full paths.
   By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button and select the location in the dialog box, that opens.
4. The JavaScript file to start the application with.
   If you are going to trace CoffeeScript, specify the path to the generated JavaScript file. The file can be generated externally or through transpilation using file watchers. For more details, see Transpiling CoffeeScript to JavaScript .
5. The application parameters: the NodeJS-specific arguments to be passed to the application start file through the process.argv array.
6. The environment variables for the NodeJS executable file, if applicable. See Run/Debug Configuration: Spy-Js for Node.js for details.
7. The configuration file with the configuration settings to apply to the tracing session.
   A configuration file is a JavaScript file with the extension `.js` or `.conf.js` that contains valid JavaScript code that meets the Spy-js configuration requirements . If PhpStorm detects files with the extension `.conf.js` in the project, these files are displayed in the drop-down list.

   Type the path to the configuration file manually or click the Browse button and choose the location in the dialog box that opens. Once specified, a configuration file is added to the drop-down list so you can get if next time from the list instead of specifying the path.
8. The trace server port. This port number must be the same as your system proxy port. If the Automatically configure system proxy check box is selected, the specified port number is automatically set for the system proxy server. Otherwise you will have to specify the value of the field in the system proxy settings manually.
   The trace server port is filled in automatically. To avoid port conflicts, it is recommended that you accept the suggested value and keep the Automatically configure system proxy check box selected.

2. To launch a tracing session, click the Run toolbar button . The Spy-js tool and the NodeJS application start. The Spy-Js Tool Window opens showing the captured events in the Trace Run tab.

Saving and loading tracing sessions

You can save an image of a tracing session and load this image at any time later. The `.json` files that store the calls and properties of the session are compressed into a `zip` archive. Upon request, when you choose Load trace, the `.json` files are extracted from the archive and loaded into Spy-js.

Note that a loaded image does not restore the session because no scripts are actually executed. All you can do is analyze the flow and properties of previously executed code.

To save an image of a tracing session

– Click the button on the Events toolbar, and then choose Save trace from the list. PhpStorm compresses all the affected `.json` files in a `zip` archive and opens the folder where the archive is saved.

To load an image of a previous tracing session

1. Initiate a tracing session of the same type as the session you are going to load the image of: a Spy-js or a Spy-js for NodeJS respectively, see Initiating a Spy-js Tracing Session above. We need to have this fake session running because otherwise Spy-js would be inactive and thus the load trace functionality would be unavailable.
   Of course you can load a trace while actually tracing an application, but in this case the currently running session will be lost after loading.

2. When the session starts, click the button on the Events toolbar and choose Load Trace from the list.

3. In the dialog box that opens, choose the location of the `zip` archive with the image of the desired session. Spy-js stops the running and shows the loaded trace in a new tab named Loaded <loaded session>.
   Note that a loaded image does not restore the session because no scripts are actually executed. All you can do is analyze the flow and properties of previously executed code.

Configuring the range of events to display

By default, the Spy-js tool captures all events on all opened Web pages, excluding https secure web sites, unless you have specified a URL address explicitly in the run configuration. The Events pane of the Spy-js tool window shows all captured events. If for some reasons you do not want to have all events captured, you can suppress capturing some of them by applying user-defined event filters. All the available filters are listed upon clicking the Capture Events button on the toolbar, the currently applied filter is marked with a tick. By default the Capture All predefined filter is applied. To stop capturing events without stopping the application, choose Mute All. The application is still running but the Events pane shows the last captured event. This is helpful if you want to analyze a script and therefore need it to be displayed in the Events pane instead of being removed as new events are captured. You can define new custom filters or add event patterns to existing filters on the fly.

Defining a new event filter

1. Click the Capture Events button on the toolbar, and then choose Edit Capture Exclusions from the list.

2. In the Spy-Js Capture Exclusions Dialog that opens, click the Add toolbar button in the left-hand pane.

3. In the right-hand pane, specify the filter name in the Exclusion name field and configure a list of exclusion rules.
   To add a rule, click , the Add Condition to Exclusion dialog box opens. Type a pattern in the Value/pattern text box, in the Condition type drop-down list specify whether the pattern should be applied to event types or script names. Note, that glob pattern matching is used. When you click OK, PhpStorm brings you to the Spy-Js Capture Exclusions Dialog .

   To edit a rule, select it in the list, click , and update the rule in the dialog box that opens. To remove a rule, select it in the list and click .

Creating exclusion rules on the fly

While navigating through the tree of already captured events in the Events pane, you may come across some events or scripts that you definitely do not want to

trace. You can create a filter as described above but in this case you will have to leave the pane. With PhpStorm, you can create an exclusion rule based on any event or script, as soon as you have detected such event or script, right from the Events pane. The rule will be either added to the currently applied filter or a new filter will be created if the current setting is Capture All.

- To add an event to an exclusion filter on the fly, select the event to exclude and choose Mute <event name> event or Mute <script name> file.

If a user-defined filter is currently applied, the new rule is added to it silently. If Capture All is currently active, the Spy-Js Capture Exclusions Dialog opens, where you can create a new filter based on the selected event or script or choose an existing filter and add the new rule to it.

Setting timestamp labels

Timestamp labels help you to analyze your code execution within a specific period of time. For example, you can set two timestamp labels and view which events were captured between them. Or on the contrary, you can locate the events that were not captured within a certain period of time although you expected them to be and thus detect performance problems.

- To set a timestamp label, choose Add Label on the context menu in the Events pane. A label Labelled at <timestamp> is added under the document node.

Synchronization and navigation between the oanes and the editor

The Events and Event Stack panes are synchronized: when you click an event or script in the Events pane, its call stack is displayed in the Event Stack pane. To have also the corresponding trace file opened in the editor, press the Autoscroll to Trace toggle button on the toolbar.

The Event Stack pane is synchronized with the editor: when you click an item in the stack tree twice, the corresponding trace file opens in the editor with the cursor positioned at the clicked function.

To synchronize the Events pane directly with the editor, press the Autoscroll to Trace toggle button on the toolbar. In this case, as soon as you click a node in the Events pane, its call stack is displayed in the Event Stack pane and the corresponding trace file is opened in the editor. With the Autoscroll to Trace mode turned on, when you navigate through the Event Stack the corresponding files are also automatically opened in the editor with the corresponding functions highlighted.

Navigating from an event or a script to the trace file

A trace file is a write-protected prettified version of the script selected in the Events pane or the script whose function is double clicked in the Event Stack pane. A trace file is named `<file name>.js.trace`. To navigate from an event or a script to the trace file, do one of the following:

- Activate automatic navigation:
  1. In the Events pane, press the Autoscroll to Trace toggle button on the toolbar.
  2. Click the required event or script.

- In the Event Stack pane, click the required item in the call stack twice or choose Jump to Trace. on the context menu. The trace file opens with the cursor positioned at the clicked function.

Navigating from an event or script to the source file

You can also navigate to the source file displayed as is, without prettifying.

- In the Event Stack pane, select the required item in the call stack and choose Jump to Source on the context menu of the selection.

If the traced site is mapped with a PhpStorm project, PhpStorm detects the corresponding local file according to the mapping and opens this file in the editor. If you are tracing a site that is not mapped to a PhpStorm project, PhpStorm opens the read-only page source, just as if you chose View Page Source in the browser.

Navigating from a function to its call

To navigate from a function to the place in the code where it was called from:

- In the Event Stack pane, select the required item in the call stack and choose Jump to Caller on the context menu of the selection.

Navigating through ECMAScript 6, TypeScript, or CoffeeScript

Spy-js supports source maps, which means that you can now jump from the Event Stack pane right to the original source code in ECMAScript 6, TypeScript or CoffeeScript and observe what code fragments were executed.

Spy-js also generates mapped trace files. These are EcmaScript 6, TypeScript, or CoffeeScript trace files with the extensions `.ts.trace`, `.coffee.trace`, or `.js.trace`. The fragments of code in these files are highlighted as if they were really executed.

1. Configure the way source maps are treated by clicking on the toolbar of the Events stack and choosing the following options on the context menu:
   - Choose Enable Source Map Look-up to enable navigation to the ECMAScript 6, TypeScript or CoffeeScript source code using the source maps generated during transpilation.
   - Choose Enable source map generation, to generate source maps for everything to map the instrumented code. Choose this option if you are going to debug the original code in Chrome Dev Tools or FireFox FireBug development tools.
   - Choose Always open source mapped trace if available, to have Spy-js try to open the mapped trace file when you invoke navigation from an event to its caller.

2. To navigate from a function to the source code, select the function in question in the Event Stack pane and choose one of the following options on the context menu of the selection:
   - To navigate to the ECMAScript 6, TypeScript or CoffeeScript source code, choose Jump to Source on the context menu of the selection.
   - To navigate to the JavaScript trace file, choose Jump to Trace.
   - To navigate to the mapped trace file (ECMAScript 6, TypeScript or CoffeeScript), choose Jump to Mapped Trace.

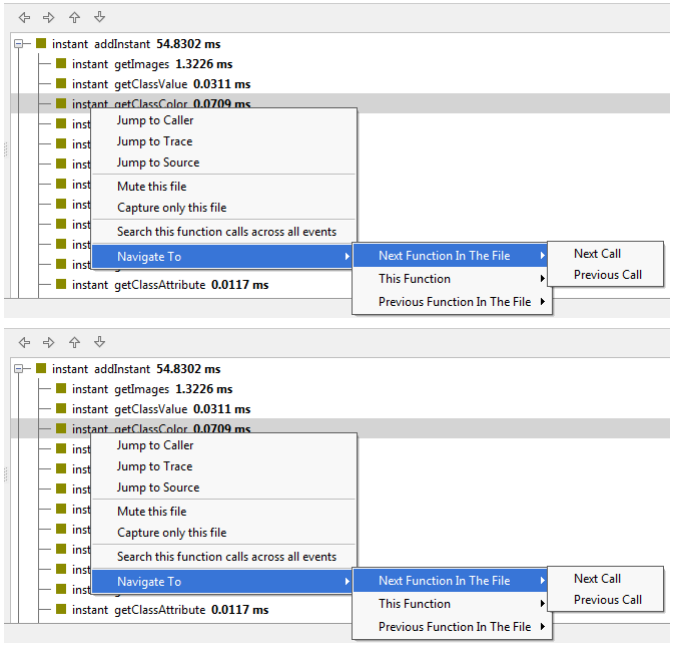
3. To navigate from a function to its call, select the function in the Event Stack and choose Jump to Caller.
   - If the Always open source mapped trace if available option is selected, the corresponding mapped trace file opens.
   - If the Always open source mapped trace if available option is not selected, the JavaScript trace file opens.

Alternatively, you can navigate to the executed JavaScript code by choosing Jump to Trace.

Advanced trace navigation

With advanced trace navigation, you can move through the whole stack based on calls and locate the functions that have not been called, that is, locate the fragments of code that have not been executed and analyze the reason for them to be skipped.

The following six actions are available: move to the next/previous call of the next/current/previous function in a trace file. The full list of actions is available from the context menu in the Event Stack pane. Moving to the next and previous calls of the selected function, to the previous call of the previous function, and to the next call of the next function are also available from the navigation toolbar of the Event Stack pane.

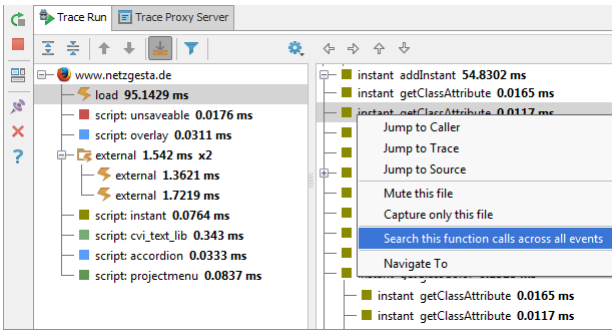


When you choose one of these actions, the cursor jumps to the call in the stack. If the Autoscroll to Trace toggle button is pressed, the corresponding trace file opens automatically with the cursor positioned at the call.

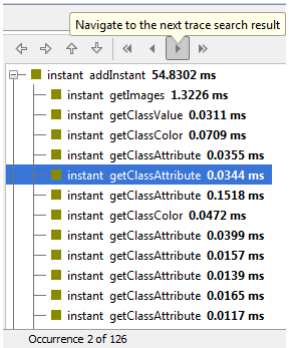
Advanced trace search

Advanced trace search lets you navigate between the calls of a function within the whole trace (across all the traced events). This means that if you are tracing 5 pages in the browser and the Events pane, accordingly, shows 5 document nodes, PhpStorm searches for the calls of the selected function under all these nodes and displays the number of found calls of the function in the Status bar.

- To invoke the search for the calls of a function in all document nodes, select the function in question in the Event Stack pane and choose Search this function calls across all events on the context menu of the selection. The number of found calls is displayed in the Status bar, and the toolbar shows four previously hidden navigation chevron buttons.



- Use the chevron buttons to navigate within the found calls:
  - To jump to the first detected call, click .
  - To jump to the last detected call, click .
  - To jump to the next detected call, click . The Status bar shows a message: `Occurrence <number> of <total number of detected calls>`
  - To jump to the previous detected call, click .



The search results are reset and the search toolbar is hidden when you invoke another advanced search or navigation.

Also keep in mind that the number of call occurrences is calculated when you choose the Search this function calls across all events option. As you analyze the detected calls, the time passes, new events are captured, and the first detected call can happen to be already removed from the stack which means that it is no longer available for navigation.

Expanding the basic completion list with runtime data (Spy-js autocompletion)

The term Spy-js autocompletion denotes expanding the basic completion list with suggestions retrieved from the runtime data. The Spy-js autocompletion functionality is available from source files for the code that has already been executed (highlighted green in the corresponding trace file).

When you position the caret at a symbol in the source file and press `Ctrl+Space`, Spy-js retrieves data from the browser or from the running NodeJS application and merges it with the basic completion list according to the following rules:

1. If an object both is present on the basic completion list and is retrieved from the runtime, the variant that provides more information about parameters, attributes, their type, etc. remains on the list.
2. Objects retrieved by Spy.js are shown on top of the list and marked with the icon. If a retrieved object is specific for a browser, the object is marked with the icon and with the icon of this browser.

Activating Spy-js autocompletion

By default, the functionality is turned off. To activate it:

– Click the button on the Events toolbar, and then choose Enable Spy-js autocomplete and magnifier from the list.

Evaluating expressions without running a debugging session (Spy-js magnification)

The term Spy-js magnification denotes evaluating expressions without actually running a debugging session. When you click the expression in question or position the caret at it and press `Ctrl+Alt+F8`, a tooltip is displayed below the expression showing the expression value. If Spy-js retrieves several values, click icon in the tooltip to expand the list of values.

The magnification functionality is available from source files for both executed and not yet executed code.

By default, the functionality is turned off.

Activating Spy-js magnification

By default, the functionality is turned off. To activate it:

– Click the button on the Events toolbar, and then choose Enable Spy-js autocomplete and magnifier from the list.

Viewing dependency diagrams

With Spy-js, you can build and examine runtime application/event diagrams for client-side and Node.js applications.

1. Generate a diagram:
   – To build a diagram with the dependencies within the entire application, select the document node and choose Show application dependency diagram on the context menu of the selection.
   – To build a diagram with the dependencies of a single event, select the required event in the Events pane and choose Show event dependency diagram.

2. Analyze the diagram:
   – The diagram is opened in a separate editor tab. The nodes in the diagram represent your project files, while the edges represent the fact that there's one or more functions in the source file that invoke functions in the target file.
   – To examine the details of a node or an edge, select the node or the edge in question and view its Details tree in a dedicated pane in the upper right-hand corner of the editor. The pane displays the connecting function combinations, along with event(s) the calls are made within and the number of calls made.

Using AngularJS

On this page:

- Overview
- Before you start
- Setting up AngularJS support in a PhpStorm project
  - Creating an empty PhpStorm project for AngularJS
  - Installing AngularJS manually in an empty project
  - Installing AngularJS in an empty project through NPM and Bower
- Generating an AngularJS application stub

Overview

PhpStorm provides integration with the AngularJS framework , versions 1 and 2. This support involves:

- AngularJS-aware code completion for `ng` directives (also including custom directives), controller and application names, and code insights for data bindings inside curly-brace expressions `{{}}` .
- AngularJS-specific navigation:
  - Between the name of a controller name in HTML and its definition in JavaScript.
  - Between `ngView` or `&routeProvider` to the template.
  - Go To Symbol navigation for entities.
- Quick documentation look-up by pressing `Ctrl+Q`, see Viewing Inline Documentation for details.
- AngularJS router diagrams .

Before you start

Make sure the NodeJS and AngularJS plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Setting up AngularJS support in a PhpStorm project

You can get AngularJS support in a PhpStorm project in two ways:

- Create an empty project and install AngularJS in it. This can be done either manually, by downloading the AngularJS framework and or by means of the Bower package manager, see Installing and Removing Bower Packages ..
  If you choose manual installation, you will need to configure AngularJS as a PhpStorm JavaScript library. If you use Bower, this will be done automatically, without any steps from your side.
- Have PhpStorm generate a stub of an AngularJS-specific project. PhpStorm sets up the AngularJS-specific project structure and generates the `bower.json` and `package.json` files with the definitions of all the necessary dependencies. See Generating an AngularJS Application Stub below on this page.

Creating an empty PhpStorm project for AngularJS

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Empty Project.
3. In the Location text box, specify the path to the project folder where the project-related files will be stored.
4. When you click Create, PhpStorm creates and opens an empty project.

Installing AngularJS manually in an empty project

1. Download the AngularJS framework at http://angularjs.org/ .
2. Open the empty project where you will use AngularJS.
3. Configure AngularJS as a PhpStorm JavaScript library, to let PhpStorm recognize AngularJS-specific structures and provide full coding assistance:
   1. Open the Settings dialog box , and click JavaScript Libraries .
   2. In the Libraries area, click the Add button.
   3. In the New Library dialog box that opens, specify the name of the library.
   4. Click the Add button + next to the list of library files and choose Attach Files or Attach Directory on the context menu, depending of whether you need separate files or an entire folder.
   5. Select the `Angular.js` or `Angular.min.js`, or an entire directory in the dialog box that opens. PhpStorm returns to the New Library dialog box where the Name read-only field shows the name of the selected files or folder.
   6. In the Type field, specify which version you have downloaded and are going to add.
      - If you added `Angular.js` , choose Debug. This version is helpful in the development environment, especially for debugging.
      - If you added the minified `Angular.min.js`, choose Release. This version is helpful in the production environment because the file size is significantly smaller.

   Learn more at Configuring JavaScript Libraries .

Installing AngularJS in an empty project through NPM and Bower

1. Open the empty project where you will use AngularJS.
2. Download and install Node.js . This runtime environment is required for two reasons:
   - The Bower package manager is started through Node.js.
   - Node Package Manager (npm), which is a part of Node.js, is also the easiest way to download Bower.

   For details on using Node.js in PhpStorm, see Node.js You can also define Node.js as an external tool, as described in the section Configuring third-party tools . This approach is helpful, when you need facilities that are missing in the plugin, for example, the possibility to pass certain parameters as wildcards.
3. Install Bower as described in Using Bower Package Manager .
4. If you are going to use the command line mode, make sure the following paths are added to the PATH variable :

1. The path to the parent folder of the Node.js executable file.
2. The path to the `npm` folder.
3. The path to the parent folder of the Bower executable file.

This enables you to launch the tools from any folder.

5. Install and enable the NodeJS and AngularJS repository plugins. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.
6. Run Bower from PhpStorm using the Bower page of the Settings dialog box.
   1. Open the Settings dialog box , and click JavaScript, then click Bower.
   2. On the Bower page that opens, the Packages area shows all the Bower-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
   3. In the Available Packages dialog box that opens, select the `AngularJS` package.
   4. Specify the installation mode:
      - To have the tool installed globally at the PhpStorm level so it can be used in any PhpStorm project, select the Options check box and type `-g` in the text box next to it.
      - To have the tool installed in the current project and thus restrict its use to this project, clear the Options check box.
   5. Optionally specify the product version and click Install Package to start installation.

Generating an AngularJS application stub

PhpStorm can generate a project stub for developing applications using AngularJS.

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose AngularJS.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. From the Version drop-down list, choose the branch https://github.com/angular/angular.js to download the project template from. By default, `master` if shown.
4. When you click Create, PhpStorm sets up the AngularJS-specific project structure and generates the `bower.json` and `package.json` files with the definitions of all the necessary dependencies.
5. Download the AngularJs dependencies that contain the AngularJS code and the tools that support development and testing.
   1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
   2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
   3. At the command line prompt, type the following command:

```
npm install
```

Learn more about the installation of dependencies in the Install Dependencies section of the `readme.md` file.

Using Angular CLI

PhpStorm provides integration with Angular CLI .

On this page:

- Before you start
- Installing Angular CLI in PhpStorm
- Generating an Angular CLI application stub
- Using Angular CLI templates

Before you start

Make sure the NodeJS and AngularJS plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Installing Angular CLI in PhpStorm

To use Angular CLI in PhpStorm projects, you need to install the `angular-cli` package globally. Do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g angular-cli
     ```

     The `-g` key makes the framework run in the global mode. Because the installation is performed through NPM, the Angular CLI framework is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the framework from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the Angular CLI framework, see https://npmjs.org/package/angular-cli .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Generating an Angular CLI application stub

1. Choose File | New Project on the main menu or click the New Project button on the Welcome screen. The New Project Dialog dialog box opens.
2. In the left-hand pane, choose Angular CLI.
3. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. In the Node Interpreter field, specify the NodeJS interpreter to use. Choose a configured interpreter from the drop-down list or choose Add to configure a new one, see Configuring Node.js Interpreters
4. When you click Create, PhpStorm sets up the AngularJS-specific project structure and generates the `bower.json` and `package.json` files with the definitions of all the necessary dependencies.

Using Angular CLI templates

The Angular CLI templates are listed on the Live Templates page of the Settings / Preferences Dialog , under the Angular2 node.

To use a template, type its abbreviation in the editor and then press Tab to expand it. Press Tab again to jump to the next edit location in the template.

Using Grunt Task Runner

PhpStorm provides integration with the Grunt JavaScript Task Runner . Grunt support involves:

- Parsing `Gruntfile.js` files, recognizing definitions of tasks and targets.
- Building trees of tasks and targets.
- Navigation between a task or a target in the tree and its definition in the `Gruntfile.js` file.
- Running and debugging tasks and targets.
- Configuring the task execution mode and output.

On this page:

- Preparing to install Grunt
- Installing Grunt
  - Installing the Grunt command line interface
  - Installing Grunt in a project
- Running Grunt tasks
  - Running Grunt tasks from the tasks tree
    - Building a tasks tree
    - Running tasks
  - Running and debugging tasks according to a run configuration
    - Creating a Grunt.js run/debug configuration
    - Running a task according to a run configuration
    - Debugging Grunt tasks
  - Running a Grunt task as a Before-Launch task
- Running Grunt tasks automatically

Preparing to install Grunt

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Grunt task runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Grunt task runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Grunt task runner and npm from any folder.

2. Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing Grunt

To use Grunt in a PhpStorm project, you need two packages:

- A globally installed `grunt-cli` package (Grunt command line interface) for executing Grunt commands.
- A `grunt` package installed in the project to build the project tasks tree and provide coding assistance while editing the `Gruntfile.js` or `Gruntfile.coffee` file. Learn more about `Gruntfile.json` at http://gruntjs.com/getting-started#preparing-a-new-grunt-project .

These packages can be installed either in the command line mode or through the PhpStorm user interface. In either case, installation is performed via npm.

Installing the Grunt command line interface

The Grunt Command Line Interface is implemented in the `grunt-cli` package which should be installed globally to make the functionality available in any PhpStorm project and from any folder. Do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g grunt-cli
     ```

     The `-g` key makes the task runner run in the global mode. Because the installation is performed through NPM, the Grunt task runner is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the task runner from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the Grunt task runner, see https://npmjs.org/package/grunt-cli .

- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing Grunt in a project

At the project level, the basic Grunt functionality is implemented in the `grunt` package. The package is responsible for building project trees of Grunt tasks and for coding assistance in editing the `Gruntfile.js` file. Do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install grunt
```

- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package.
  4. Optionally specify the product version and click Install Package to start installation.

Running Grunt tasks

Grunt tasks are launched in three ways:

- From tasks trees in the dedicated Grunt Tool Window which opens when you invoke Grunt from a `Gruntfile.js` file. As soon as you invoke Grunt, it starts building a tree of tasks according to the `Gruntfile.js` on which it was invoked. If a task has targets , the task is displayed as a node and the targets are listed under it.
- According to a dedicated run configuration, see Run/Debug Configuration: Grunt.Js .
- As a before-launch task, from another run configuration.

The result of executing a task is displayed in the Run tool window . The tool window shows the Grunt output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed task is displayed on the title bar of the tool window.

Running Grunt tasks from the tasks tree

Running Grunt tasks right from the tasks tree is easy and fast, the only disadvantage of this approach is that such important additional options as force execution and verbose mode are not available in this mode. As a result, you cannot have PhpStorm, for example, ignore warnings or provide a detailed log.

Building a tasks tree

- If the Grunt tool window is not opened yet, do one of the following:
  - Select the required `Gruntfile.js` file in the Project tool window and choose Show Grunt Tasks on the context menu of the selection.
  - Open the required `Gruntfile.js` file in the editor and choose Show Grunt Tasks on the context menu of the editor.

  In either case, the Grunt tool window opens showing the tasks tree built according to the selected or opened `Gruntfile.js` file.

- If the Grunt tool window is already opened, click on the toolbar and choose the required `Gruntfile.js` file from the list. PhpStorm adds a new node and builds a tasks tree under it. The title of the node shows the path to the `Gruntfile.js` file according to which the tree is built.

- To re-build a tree, switch to the required node and click on the toolbar.

Running tasks

- To run a task or a target, do one of the following:
  - To run the task, select the root node in the tree and choose on the context menu of the selection.
  - To navigate to the definition of a task or target, select the required task or target in the tree and choose on the context menu of the selection.

  The task or target execution output will be displayed in the Run tool window . The name of the target is shown in the format `<task name>:<target name>` . The tool window shows the Grunt output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed task is displayed on the title bar of the tool window.

- To run several tasks or targets, use the multiselect mode: hold `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys and select the required tasks or targets, then choose Run on the context menu of the selection.

Running and debugging tasks according to a run configuration

Besides using temporary run configurations that PhpStorm creates automatically, you can create and launch your own Grunt.js run configurations. For details about run configurations, see Run/Debug Configuration and Creating and Editing Run/Debug Configurations .

Creating a Grunt.js run/debug configuration

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.

2. Click the Add New Configuration button + on the toolbar and select Grunt.js from the pop-up list.

3. In the Run/Debug Configuration: Grunt.Js dialog box that opens, specify the name of the run configuration, the tasks to run (use blank spaces as separators), the location of the `Gruntfile.js` file to retrieve the definitions of the tasks from, and the path to the `Grunt` package installed locally, under the project root. Specify the location of the Node executable file and the NodeJS-specific options to be passed to this executable file, see Node parameters for details.

If applicable, specify the environment variables for the NodeJS executable file.

To have Grunt ignore warnings and continue executing the launched task until the task is completed successfully or an error occurs, select the --f check box. Otherwise, the task execution is stopped by the first reported warning. To have the verbose mode applied and thus have a full detailed log of a task execution displayed, select the --v check box.

Running a task according to a run configuration

Select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button. The output is displayed in the Run tool window .

Debugging Grunt tasks

1. Create a Grunt.js run/debug configuration, see above .
2. Open the `Gruntfile.js` file in the editor and set the breakpoints in it where necessary.
3. To start a debugging session, select the required debug configuration from the list on the main tool bar and then choose Run | Debug <configuration name> on the main menu or click the Debug toolbar button.
4. In the Debug tool window that opens, analyze the suspended task execution, step through the task, etc. as described in Examining Suspended Program and Stepping Through the Program .

Running a Grunt task as a Before-Launch task

1. Open the Run/Debug Configurations dialog by choosing Run | Edit Configurations on the main menu, and select the required configuration from the list or create it anew by clicking + and choosing the relevant run configuration type.
2. In the dialog box that opens, click + in the Before launch area and choose Run Grunt task from the drop-down list.
3. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
   Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.

Running Grunt tasks automatically

If you have some tasks or targets that you run on a regular basis, you can add the corresponding run configurations to a list of startup tasks. The tasks will be executed automatically on the project start-up.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Startup Tasks under Tools.
2. On the Startup Tasks page that opens, click + on the toolbar.
3. From the drop-down list, choose the required Grunt run configuration. The configuration is added to the list.
   If no applicable configuration is available in the project, click + and choose Edit Configurations. Then define a configuration with the required settings in the Run/Debug Configuration: Grunt.Js page that opens. When you save the new configuration it is automatically added to the list of startup tasks.

Using Gulp Task Runner

PhpStorm provides integration with the Gulp.js Task Runner . Gulp.js support involves:

- Parsing `Gulpfile.js` files, recognizing definitions of tasks.
- Building trees of tasks.
- Navigation between a task in the tree and its definition in the `Gulpfile.js` file.
- Running and debugging tasks.
- Configuring the task execution mode and output.
- Creating a Gulp.js run configuration automatically.

On this page:

- Preparing to install Gulp.js
- Installing Gulp.js
  - Installing Gulp.js in the command line mode
  - Installing Gulp.js through the PhpStorm interface
- Running Gulp.js tasks
  - Running Gulp.js tasks from the tasks tree
    - Building a tasks tree
    - Running a task
  - Running tasks from Gulpfile.js
  - Running and debugging tasks according to a run configuration
    - Creating a Gulp.js run configuration
    - Running a task according to a run configuration
    - Debugging Gulp tasks
  - Running a Gulp task as a Before-Launch task
- Running Gulp.js tasks automatically

Preparing to install Gulp.js

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Gulp.js task runner is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Gulp.js task runner.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Gulp.js task runner and npm from any folder.

2. Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing Gulp.js

To run Gulp in a PhpStorm project you need a Gulp package installed locally, under the project root.

You can additionally have a global installation of Gulp. In this case, the path to the Gulp installation will be added to the `PATH` environment variable. This will allow you to invoke Gulp from any folder when working in the command line mode.

The basic Gulp.js functionality is implemented in the `gulp.js` package. The package is responsible for building project trees of Gulp tasks and for coding assistance in editing the `Gulpfile.js` file. The `gulp` package can be installed either in the command line mode or through the PhpStorm user interface. In either case, installation is performed via npm.

Installing Gulp.js in the command line mode

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).

2. Do one of the following:
   - To install Gulp.js in the current project, switch to the project root folder and type the following command at the command line prompt:

     ```
     npm install gulp
     ```

   - To install Gulp.js globally, type:

     ```
     npm install --global gulp
     ```

Installing Gulp.js through the PhpStorm interface

1. Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box: Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.

2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.

3. In the Available Packages dialog box that opens, select the `gulp` package.
   Optionally:
   - Specify the version to install.
   - To have the package installed globally, select the Options check box and type `--global` in the text box.

4. Click Install Package to start installation.

Running Gulp.js tasks

Gulp.js tasks can be run either from the tasks tree in the dedicated Gulp Tool Window , from the `Gulpfile.js` file, by launching a Gulp.js run configuration , or as aa before-launch task from another run configuration.

Running Gulp.js tasks from the tasks tree

Gulp.js starts building a tasks tree as soon as you invoke Gulp.js by choosing Show Gulp Tasks on the context menu of a `Gulpfile.js` in the Project tool window or of a `Gulpfile.js` opened in the editor. The tree is built according to the `Gulpfile.js` file on which Gulp.js was invoked. If you have several `Gulpfile.js` files in your project, you can build a separate tasks tree for each of them and run tasks without abandoning the previously built tasks trees. Each tree is shown under a separate node.

Technically, PhpStorm invokes Gulp.js and processes `Gulpfile.js` according to the default Gulp.js run configuration. However this is done silently and does not require any steps from your side.

Building a tasks tree

- If the Gulp tool window is not opened yet, do one of the following:
  - Select the required `Gulpfile.js` file in the Project tool window and choose Show Gulp Tasks on the context menu of the selection.
  - Open the required `Gulpfile.js` file in the editor and choose Show Gulp Tasks on the context menu of the editor.

  In either case, the Gulp tool window opens showing the tasks tree built according to the selected or opened `Gulp.js` file.

- If the Gulp tool window is already opened, click on the toolbar and choose the required `Gulpfile.js` file from the list. PhpStorm adds a new node and builds a tasks tree under it. The title of the node shows the path to the `Gulpfile.js` file according to which the tree is built.

- To re-build a tree, switch to the required node and click on the toolbar.

If your `Gulpfile.js` is written in ECMA6 , by default PhpStorm does not recognize this format and fails to build a tasks tree. To solve this problem, update the default Gulp.js run configuration:

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.

2. Under the Defaults node, click Gulp.js.

3. In the Run/Debug Configuration: Gulp.Js dialog box that opens, type `--harmony` in the Node options text box and click OK.

Running a task

- To run a Gulp.js task from a tree of tasks, do one of the following:
  - To run the task, select the root node in the tree and choose on the context menu of the selection.
  - To navigate to the definition of a task, select the required task in the tree and choose on the context menu of the selection.

  The task execution output will be displayed in the Run tool window . The tool window shows the Gulp.js output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed task is displayed on the title bar of the tool window.

- To run several tasks, use the multiselect mode: hold `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys and select the required tasks, then choose Run on the context menu of the selection.

Running tasks from Gulpfile.js

You can run tasks right from the `Gulpfile.js` file opened in the editor without previously building a tree of tasks.

- Position the cursor at the definition of the task to run and choose Run <task name> on the context menu of the selection.
  PhpStorm creates and launches a temporary run configuration with the name of the selected task.

  The task execution output will be displayed in the Run tool window . The tool window shows the Gulp.js output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed task is displayed on the title bar of the tool window.

- To save an automatically created temporary run configuration, position the cursor at the definition of the task for which it was created and choose Save <task name> on the context menu of the selection. Learn more in Creating and Saving Temporary Run/Debug Configurations .

Running and debugging tasks according to a run configuration

Besides using temporary run configurations that PhpStorm creates automatically, you can create and launch your own Gulp.js run configurations. For details about run configurations, see Run/Debug Configuration and Creating and Editing Run/Debug Configurations .

Creating a Gulp.js run configuration

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.

2. Click the Add New Configuration button on the toolbar and select Gulp.js from the pop-up list.

3. In the Run/Debug Configuration: Gulp.Js dialog box that opens, specify the name of the run configuration, the tasks to run (use blank spaces as separators), the location of the `Gulpfile.js` file to retrieve the definitions of the tasks from, and the path to the `Gulp` package installed locally, under the project root.

Specify the location of the Node executable file and the NodeJS-specific options to be passed to this executable file, see Node parameters for details.

If applicable, specify the environment variables for the NodeJS executable file.

If applicable, in the Arguments field, specify the arguments for tasks to be executed with. Use the following format:

```
--<parameter_name> <parameter_value>
```

For example: `--env development`.
For details about passing task arguments, see https://github.com/gulpjs/gulp/blob/master/docs/recipes/pass-arguments-from-cli.md

Running a task according to a run configuration

– Select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button. The output is displayed in the Run tool window.

Debugging Gulp tasks

1. Create a Gulp.js run/debug configuration, see above.
2. Open the `Gulpfile.js` file in the editor and set the breakpoints in it where necessary.
3. To start a debugging session, select the required debug configuration from the list on the main tool bar and then choose Run | Debug <configuration name> on the main menu or click the Debug toolbar button.
4. In the Debug tool window that opens, analyze the suspended task execution, step through the task, etc. as described in Examining Suspended Program and Stepping Through the Program.

Running a Gulp task as a Before-Launch task

1. Open the Run/Debug Configurations dialog by choosing Run | Edit Configurations on the main menu, and select the required configuration from the list or create it anew by clicking + and choosing the relevant run configuration type.
2. In the dialog box that opens, click + in the Before launch area and choose Run Gulp task from the drop-down list.
3. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
   Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.

Running Gulp.js tasks automatically

If you have some tasks that you run on a regular basis, you can add the corresponding run configurations to a list of startup tasks. The tasks will be executed automatically on the project start-up.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Startup Tasks under Tools.
2. On the Startup Tasks page that opens, click + on the toolbar.
3. From the drop-down list, choose the required Gulp.js run configuration. The configuration is added to the list.
   If no applicable configuration is available in the project, click + and choose Edit Configurations. Then define a configuration with the required settings in the Run/Debug Configuration: Gulp.Js page that opens. When you save the new configuration it is automatically added to the list of startup tasks.

Using JavaScript Code Quality Tools

PhpStorm provides facilities to run JavaScript-specific code quality inspections through integration with the following code verification tools:

- JSLint . The tool is bundled with PhpStorm.
- JSHint . PhpStorm comes bundled with the default version of the tool, but you can have an alternative version downloaded.
- Closure Linter . The tool is not bundled with PhpStorm, so you need to download it yourself.
- JSCS . The tool is not bundled with PhpStorm, but you can install it from the PhpStorm interface using the Node Package Manager (npm).
- ESLint . The tool is not bundled with PhpStorm, but you can install it from the PhpStorm interface using the Node Package Manager (npm).

All these tools register themselves as PhpStorm code inspections: they check JavaScript code for most common mistakes and discrepancies without running the application. When a tool is activated, it launches automatically on the edited JavaScript file. Discrepancies are highlighted and reported in pop-up information windows, a pop-up window appears when you hover the mouse pointer over a stripe in the Validation sidebar. You can also press `Alt+Enter` to examine errors and apply suggested quick fixes. Learn more about inspections and intention actions at Code Inspection and Intention Actions .

On this page:

- Enabling and configuring the built-in JSLint tool
- Enabling and configuring JSHint
- Enabling and configuring Closure Linter
- Using JSCS
- Using ESLint

Enabling and configuring the built-in JSLint tool

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the JavaScript node under Languages&Frameworks, and then click JSLint under Code Quality Tools.
2. On the JSLint page that opens, select the Enable check box. After that all the controls in the page become available.
3. Define the set of common mistakes to check the code for. To enable a validation, select the check box next to it.

Enabling and configuring JSHint

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the JavaScript node under Languages&Frameworks, and then click JSHint under Code Quality Tools.
2. On the JSHint page that opens, select the Enable check box. After that all the controls in the page become available.
3. From the Version drop-down list, choose the version of the tool to apply.
   PhpStorm comes bundled with version 1.0.0, which is used by default. PhpStorm provides the ability to download another version, which is not bundled. Actually, the alternative version is downloaded only once, whereupon it is available without download.
4. Configure the behaviour of JSHint:
   - To have the code verified according to the rules from a previously created configuration file, select the Use config files check box.
     A configuration file is a JSON file with the extension `.jshintrc` that specifies which JSHint options should be enabled or disabled. PhpStorm will look for a `.jshintrc` file in the working directory. If the search fails, PhpStorm will search in the parent folder, then again in the parent folder. The process is repeated until PhpStorm finds a `.jshintrc` or reaches the project root. To have PhpStorm still run verification in this case, specify the default configuration file to use.

     Starting with PhpStorm version 6.0.1, the globals setting in `.jshintrc` is supported. For earlier versions, a Predefined (, separated)" tree node is displayed in the bottom of the tree in the UI.

   - To configure verification manually, clear the check box and define the set of common mistakes to check the code for in the Options area. The controls in the area fall into two groups:
     - Enforcing options: select the check boxes in this group to enable very strict behaviour of the verification tool and thus allow only safe JavaScript.
     - Relaxing options: select/clear the check boxes in this area to suppress warnings when certain types of discrepancies are detected.
     - Environments: select/clear these check boxes to specify for which environments you want global properties predefined.

     To enable a validation, select the check box next to it.

Enabling and configuring Closure Linter

1. Download and install the Closure Linter tool .
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the JavaScript node under Languages&Frameworks, and then click Closure Linter under Code Quality Tools.
3. On the Closure Linter page that opens, select the Enable check box. After that all the controls in the page become available.
4. Specify the path to the Closure Linter executable file:
   - `<Python_home>\Scripts\jslint.exe` for Windows
   - `/usr/local/bin/gjslint` for Linux and OS X
5. Specify the path to the previously created configuration file.

Note that the behaviour of Closure Linter may slightly differ depending on whether it is run from PhpStorm or in the command line mode if a configuration file is parsed before the `gjslint` process started and all the parsed options are passed as arguments to the `gjslint` process. When Closure Linter is running inside PhpStorm, the `gjslint` process is launched in the following way:

```
/path/to/gjslint --flagfile /path/to/config_file --recurse=no /path/to/user_source_file.js
```

Therefore, the following command fails if the configuration file has a space instead of a `=`:

```
--jslint_error indentation
```

Using JSCS

The JSCS tool is run through NodeJS, therefore make sure the NodeJS framework is downloaded and installed on your computer. The framework also contains the Node Package Manager(npm) through which JSCS is installed.

Integration with NodeJS and NPM is supported through the NodeJs plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

To install JSCS:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `jscs` package and click Install Package.
   Learn more about installing tools through NPM in Installing and Removing External Software Using Node Package Manager .

To activate and configure JSCS:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.
   Expand the JavaScript node under Languages&Frameworks, and then click JSCS under Code Quality Tools.
2. Select the Enable check box to activate JSCS. After that the controls in the dialog box become available.
3. In the Node Interpreter field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the button and select the location in the dialog box, that opens.
4. In the JSCS Package field, specify the location of the `jscs` package installed in the current project, see Installing JSCS .
5. Appoint the configuration to use.
   By default, PhpStorm first looks for a `jscsConfig` property in the `package.json` file of the current project. If no such property is found, PhpStorm looks for a `.jscsrc` or a `.jscs.json` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. Accordingly, you have to define the configuration to apply either as a `jscsConfig` property in the `package.json` file or in a `.jscsrc` or a `.jscs.json` configuration file, or in a custom JSON configuration file.

   You can also apply a predefined set of rules, either independently or in combination with a configuration file. In the latter case, the rules from the configuration file override the predefined rules.
   - To have PhpStorm look for a `jscsConfig` property in the `package.json` file or for a `.jscsrc` or a `.jscs.json` file, choose the Search for config(s) option.
   - To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the button and select the relevant file from the dialog box that opens.
   - To have a predefined set or rules applied, choose the desired set from the Code Style Preset drop-down list.
6. If necessary, from the Code Style Preset drop-down list, choose the set of predefined rules associated with the code style you use.

Using ESLint

The ESLint tool is run through NodeJS, therefore make sure the NodeJS framework is downloaded and installed on your computer. The framework also contains the Node Package Manager(npm) through which JSCS is installed.

Integration with NodeJS and NPM is supported through the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

To install ESLint:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `eslint` package and click Install Package.
   Learn more about installing tools through NPM in Installing and Removing External Software Using Node Package Manager .

To activate and configure ESLint:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.
   Expand the JavaScript node under Languages&Frameworks, and then click ESLint under Code Quality Tools.
2. Select the Enable check box to activate ESLint. After that the controls in the dialog box become available.
3. In the Node Interpreter field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the button and select the location in the dialog box, that opens.
4. In the ESLint Package field, specify the location of the `eslint` package installed in the current project, see Installing ESLint .
5. Appoint the configuration to use.
   By default, PhpStorm first looks for a `.eslintrc` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. If no `.eslintrc` file is found, ESLint uses its default embedded configuration file. Accordingly, you have to define the configuration to apply either in a `.eslintrc` configuration file, or in a custom JSON configuration file, or rely on the default embedded configuration.
   - To have PhpStorm look for a `.eslintrc` file, choose the Search for .eslintrc option. If no `.eslintrc` file is found, the default embedded configuration file will be used.
   - To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the button and select the relevant file from the dialog box that opens.
6. If necessary, in the Additional Rules Directory field, specify the location of the files with additional code verification rules. These rules will be applied after the rules from `.eslintrc` or the above specified custom configuration file and accordingly will override them.
7. If necessary, in the Extra ESLint Options field, specify additional command line options to run ESLint with using spaces as separators. See ESLint Command Line Interface Options for details.

Using Meteor

PhpStorm provides integration with the Meteor framework . Meteor support in PhpStorm involves:

- Automatic recognition of Meteor projects by detecting the `.meteor` folder and excluding the `.meteor/local` folder from project.
- Attaching the predefine Meteor library to the project automatically.
- Coding assistance: syntax highlighting, code completion, resolving references.
- Support of Spacebars via Handlebars with completion for if and each directives. There is also support for navigation between JavaScript source code and templates, allowing for easy navigation by using `Ctrl+B` to go to Declaration .
- Running and debugging the server and the client sides of applications.

On this page:

- Before you start
- Installing Meteor
- Preparing to develop a Meteor application
  - Generating a Meteor application stub
  - Enabling Meteor integration in an existing project
  - Importing Meteor packages
- Running and debugging a Meteor application
  - Creating a Meteor run configuration
  - Configuring the update policy in the debugging mode
    - Configuring the update policy for the server side code
    - Configuring the update policy for the client side code
  - Running a Meteor application
  - Debugging a Meteor application

Before you start

Make sure the Meteor and the Handlebars/Mustache plugins are activated. The Handlebars/Mustache enables PhpStorm to recognize Spacebars templates that are an extension of the Handlebars/Mustache templates. As a side effect, HTML files in Meteor projects are marked with the Handlebars/Mustache icon. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Installing Meteor

The installation procedure depends on the operating system you are using:

1. To install Meteor on Windows, download the `LaunchMeteor.exe` installer at https://github.com/meteor/meteor/wiki/Preview-of-Meteor-on-Windows .
2. To install Meteor on OSX (OSX 10.6 or higher is required) or Linux, type the following command:

```
$ curl https://install.meteor.com | /bin/sh
```

Learn more at http://docs.meteor.com/ .

Preparing to develop a Meteor application

To start your development, you need a PhpStorm project with the Meteor-specific structure. You can have an application stub that meets these requirements generated automatically or open an existing Meteor project in PhpStorm and configure Meteor support in it.

Generating a Meteor application stub

1. In the left-hand pane, choose Meteor JS App.
2. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the location of the Meteor executable file (see Installing Meteor).
   3. From the Template drop-down list, choose the sample to generate. To have a basic project structure generated, choose the Default option.
   4. In the Filename text box, type the name for the mutually related `.js` , `.html` , and `.css` files that will be generated. The text box is available only if the Default sample type is selected from the Template drop-dow list.

When you click Create, PhpStorm generates a skeleton of a Meteor application, including an HTML file, a JavaScript file, a CSS file, and a `.meteor` folder with subfolders. The `.meteor/local` folder, which is intended for storing the built application, is automatically marked as excluded and is not involved in indexing.

By default, excluded files are shown in the project tree. To have the `.meteor/local` folder hidden, click the button on the toolbar of the Project tool window and remove a tick next to the Show Excluded Files option.

PhpStorm also automatically attaches the predefined Meteor library to the project, thus enabling syntax highlighting, resolving references, and code completion. See Configuring JavaScript Libraries for details.

Meteor uses Spacebars templates that are an extension of the Handlebars/Mustache templates. PhpStorm recognizes Spacebars templates, but as a side effect marks HTML files in Meteor projects with the Handlebars/Mustache icon.

Enabling Meteor integration in an existing project

PhpStorm automatically detects Meteor projects by locating the `.meteor` folder. The `.meteor/local` folder, which is intended for storing the built application, is automatically marked as excluded and is not involved in indexing.

1. Open the desired Meteor project in PhpStorm by choosing File | Open on the main menu or clicking Open on the Welcome Screen.
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Meteor under JavaScript.
3. On the Meteor page that opens:

1. Check the location of the Meteor executable file or specify the path to it if PhpStorm has not detected the executable file automatically.
2. To have PhpStorm involve the `.meteor/local` folder and its contents in indexing, clear the Automatically exclude ".meteor/local" directory on open project check box.
   By default, the check box is selected and the `.meteor/local` folder, which is intended for storing the built application, is automatically marked as excluded and is not involved in indexing.
   By default, excluded files are shown in the project tree. To have the `.meteor/local` folder hidden, click the button on the toolbar of the Project tool window and remove a tick next to the Show Excluded Files option.
3. Make sure the Automatically import Meteor packages as external library check box is selected.
   - When the check box is selected, PhpStorm automatically imports the external packages from the `meteor/packages` file. As a result, PhpStorm provides full range coding assistance: resolves references to Meteor built-in functions, for example, `check(true)`, and to functions from third-party packages, provides proper syntax and error highlighting, supports debugging with source maps, etc.
   - When this check box is cleared, PhpStorm does not automatically import the external packages from the `meteor/packages` file. As a result no coding assistance is provided. To improve the situation, open the `meteor/packages` file in the editor and click the Import packages as library link or run the `meteor --update` command.

   By default, the check box is selected.
4. Click Apply to save the settings without leaving the dialog box.

4. Check that PhpStorm has attached the Meteor library to the project:
   1. Click Libraries under JavaScript.
   2. On the JavaScript. Libraries page that opens, make sure the check box next to the Meteor project library in the Libraries list is selected.
   3. Click OK.

   Alternatively, choose Use JavaScript Library on the context menu of the editor and make sure a tick is set next to Meteor project library.

By default, excluded files are shown in the project tree. To have the `.meteor/local` folder hidden, click the button on the toolbar of the Project tool window and remove a tick next to the Show Excluded Files option.

PhpStorm also automatically attaches the predefined Meteor library to the project, thus enabling syntax highlighting, resolving references, and code completion. See Configuring JavaScript Libraries for details.

Importing Meteor packages

Besides the predefined Meteor library that ensures basic Meteor-specific coding assistance, you can download additional packages that are defined in the `.meteor/local/packages` file.

1. Open the `.meteor/local/packages` file in the editor.
2. Click the Import Meteor Packages link in the upper right-hand corner of the screen.
3. In the dialog box that opens, specify the packages to download depending on the type of the application you are going to develop in your project.
   - Client
   - Server
   - Cordova: choose this option to import the packages that support development of Meteor applications for iOS and Android, see Meteor Cordova Phonegap Integration for details.

Running and debugging a Meteor application

With PhpStorm, you can debug both the client-side and the server-side of Meteor JavaScript code simultaneously using the PhpStorm debugger. The debugger also pauses at the breakpoints set in the sources stored in the `/packages` folder. This functionality is supported both for the client side and for the server side code.

Meteor applications are launched only through a dedicated run configuration. Technically, several Meteor projects that implement different applications can be combined within one single PhpStorm project. To run and debug these applications independently, create a separate run configuration for each of them with the relevant working directory. To avoid port conflicts, these run configurations should use different ports. In the Program Arguments field, specify a separate port for each run configuration in the format `--port=<port_number>`. The results are shown in the console and/or in the browser depending on the settings specified in the Browser / Live Edit tab, see Creating a run configuration below.

In the debugging mode, PhpStorm provides the Live Edit functionality which supports both automatic and manual upload of updated files on the server side. For the client side, Live Edit does not provide any way to apply the changes. For information on enabling and configuring the Live Edit functionality, see Live Editing of HTML, CSS, and JavaScript .

Creating a Meteor run configuration

Technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated Meteor run configuration.

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button, and choose Meteor on the context menu.

2. In the Configuration tab of the Run/Debug Configuration: Meteor dialog box that opens, specify the name of the configuration and the location of the Meteor executable file according to the installation (see Installing Meteor).
   Optionally, in the Program Arguments field, specify the command line additional parameters to be passed to the executable file on start up, if applicable. These can be, for example, `--dev`, `--test`, or `--prod` to indicate the environment in which the application is running (development, test, or production environments) so different resources are loaded on start up.

   In the Working Directory field, specify the folder under which the application files to run are stored. This folder must contain a `.meteor` folder in the root to be treated a Meteor project. By default, the field shows the path to the PhpStorm project root.

   Technically, several Meteor projects that implement different applications can be combined within one single PhpStorm project. To run and debug these applications independently, create a separate run configuration for each of them with the relevant working directory. To avoid port conflicts, these run configurations should use different ports. In the Program Arguments field, specify a separate port for each run configuration in the format `--port=<port_number>`.

3. Switch to the Browser / Live Edit tab and configure the behaviour of the browser:

1. To view the results of the client-side code execution during running and debugging, select the After Launch check box and choose the browser to open from the drop-down list.
2. In the text box below, specify the URL address to open the application in. The default value is `http://localhost:3000`.
3. To enable debugging the client-side code in the browser, select the With JavaScript Debugger check box.
4. Click OK to save the run/debug configuration.

Configuring the update policy in the debugging mode

In the debugging mode, PhpStorm provides the Live Edit functionality which supports both automatic and manual upload of updated files on the server side. For the client side, Live Edit does not provide any way to apply the changes.

For information on enabling and configuring the Live Edit functionality, see Live Editing of HTML, CSS, and JavaScript .

The client-side code can be updated through the native Meteor hot code pushes functionality .

Configuring the update policy for the server side code

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Debugger node under Build, Execution, Deployment, and then click Live Edit. The Live Edit page that opens.

2. In the Update area, configure the way changes made to the code during a debugging session are applied. This functionality is available only for the server-side code. To have the client side code updated, select the Enable Meteor 'Hot code push' check box on the Meteor page of the Settings dialog box.
   Note that an update will be performed only if none of the modified files have any syntax errors.
   - Auto in (ms): Choose this option to have the changes applied automatically at certain time interval and specify the delay in ms in the text box. This policy is not available for the client-side code of Meteor applications.
   - Manual: Choose this option to apply the changes manually by clicking the Update <run configuration name> JavaScript button or the Update <run configuration name> button on the toolbar of the Debug tool window.
     It is recommended that you choose this option because applying changes to the client-side code is not supported.
   - Restart if hotswap fails:
     With changes in HTML, CSS, and JavaScript on the client side, the contents of a Web page in the browser are updated without reloading. For NodeJS or Meteor applications, PhpStorm first tries to update the application incorporating the changes without restarting the NodeJS server.
     - Select this check box to have PhpStorm try restarting the server if the changes cannot be applied automatically. After the restart, PhpStorm brings you to the execution point where the problem took place.
       If even with this option chosen automatic upload still fails, tou will have to restart the server manually by clicking the Rerun <run configuration name> button.
     - When the check box is cleared, PhpStorm just displays a pop-up window informing you about the failure and suggesting you to restart the server manually.

Configuring the update policy for the client side code

The client-side code can be updated through the native Meteor hot code pushes functionality . The functionality is enabled by default, to activate it:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Meteor under JavaScript.

2. On the Meteor page that opens, select the Enable Meteor 'Hot code push' check box.

Running a Meteor application

- To run an application, select the required run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button.
  If in the run configuration the browser is configured to open automatically upon application start, the browser opens showing the result of application execution. You can view the same result by opening the browser of your choice manually at the URL specified in the run configuration.

Debugging a Meteor application

1. Set the breakpoints in the code, where necessary.

2. To initiate a debugging session, select the required debug configuration from the list on the main tool bar and then choose Run | Debug <configuration name> on the main menu or click the Debug toolbar button. The Debug tool window opens showing two tabs: one for debugging the server-side code marked with and the other for debugging the client-side code marked with.

3. Explore the suspended program and step through the program .

4. Apply the changes to the application depending on the update policy defined in the run configuration:
   - To apply the changes to the client-side code, switch to the tab and click the Update <run configuration name> JavaScript button on the toolbar.
   - The changes to the server-side code are applied depending on the update policy defined in the run configuration:
     - If you have chosen manual upload, switch to the tab and click the Update <run configuration name> button on the toolbar.
     - If you have chosen automatic upload, updates to the server-side code are applied automatically. If automatic upload fails, restart the application by clicking Rerun <run configuration name> button. Choosing the Restart if hotswap fails option on the Live Edit page may help: PhpStorm will attempt to restart the server automatically.

Using PhoneGap/Cordova

In PhpStorm, you can develop applications intended for running on various mobile platforms, including Android , using the PhoneGap , Apache Cordova , and Ionic frameworks.

On this page:

- Before you start
- Installing PhoneGap/Cordova/Ionic
  - Installing PhoneGap in the command line mode
  - Installing PhoneGap/Cordova/Ionic through the PhpStorm interface
- Preparing to use PhoneGap/Cordova/Ionic in a project
  - Generating a PhoneGap/Cordova/Ionic application stub
  - Enabling PhoneGap/Cordova/Ionic integration in an existing project
- Creating and launching a PhoneGap/Cordova/Ionic run configuration

Before you start

1. Make sure the PhoneGap/Cordova and the NodeJS plugins are enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.
2. Download and install Node.js because NPM, which is a part of the framework, is also the easiest way to download PhoneGap and Cordova.
3. Download and install an emulator tool. These tools are specific for the target platform and the operating system you use:
   - http://cordova.apache.org/docs/en/latest/guide/platforms/ios/index.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/blackberry10/home.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/osx/index.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/ubuntu/index.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/win8/index.html
   - http://cordova.apache.org/docs/en/latest/guide/platforms/wp8/home.html

Installing PhoneGap/Cordova/Ionic

To use PhoneGap/Cordova, you need a `phonegap` , or `cordova` , or `ionic` package. The package can be installed either in the command line mode or through the PhpStorm user interface. In either case, installation is performed via npm.

Installing PhoneGap in the command line mode

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Type one of the following commands at the command line prompt:

```
npm install -- global phonegap
npm install -- global cordova
npm install -- global ionic
```

   With the `--global` option, the package will be installed globally so it is accessible from all your PhpStorm projects.

Installing PhoneGap/Cordova/Ionic through the PhpStorm interface

1. Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box: Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `phonegap` , or `cordova` , or `ionic` package, depending on your preferences and workflow. Optionally:
   - Specify the version to install.
   - To have the package installed globally so it is accessible from all your PhpStorm projects, select the Options check box and type `--global` in the text box.
4. Click Install Package to start installation.

Preparing to use PhoneGap/Cordova/Ionic in a project

To start your development, you need a PhpStorm project with the PhoneGap/Cordova/Ionic-specific structure. You can have an application stub that meets these requirements generated automatically or open an existing PhoneGap/Cordova/Ionic project in PhpStorm and configure PhoneGap/Cordova/Ionic support in it.

Generating a PhoneGap/Cordova/Ionic application stub

1. In the left-hand pane, choose PhoneGap/Cordova App.
2. In the right-hand pane:
   1. In the Location text box, specify the path to the project folder where the project-related files will be stored.
   2. Specify the location of the executable file `phonegap.cmd` , or `cordova.cmd` , or `ionic.cmd` (see Installing PhoneGap/Cordova/Ionic).

When you click Create, PhpStorm generates a skeleton of a PhoneGap/Cordova/Ionic application with the framework-specific structure.

Enabling PhoneGap/Cordova/Ionic integration in an existing project

1. Open the desired PhoneGap/Cordova/Ionic project in PhpStorm by choosing File | Open on the main menu or clicking Open on the Welcome Screen.
2. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click PhoneGap/Cordova under JavaScript.
3. On the PhoneGap/Cordova page that opens:
   1. Check the location of the executable file `phonegap.cmd`, or `cordova.cmd`, or `ionic.cmd` or specify the path to it if PhpStorm has not detected the executable file automatically.
      PhpStorm detects the installed version and displays it in the PhoneGap/Cordova Version read-only field,
   2. In the PhoneGap/Cordova Working Directory field, specify the folder under which the PhoneGap/Cordova/Ionic application files to run are stored.
   3. In the Plugins area, configure a list of plugins to use in your development by installing required packages. The list shows all the PhoneGap/Cordova/Ionic plugins that are currently installed on your computer, both at the global and at the project level.
      - To install a plugin, click the Install button +. In the Available Packages dialog box that opens, select the required package.
        To have the plugin installed globally so it is accessible from all your PhpStorm projects, select the Options check box and type `--global` in the text box. Click Install Package.
      - To remove a plugin, select it in the list and click the Uninstall button —.
      - To upgrade a plugin to the latest available version, select the plugin in the list and click the Upgrade button ↑.

      See Apache Cordova Plugins and PhoneGap Plugins for information about plugins and their use.

Creating and launching a PhoneGap/Cordova/Ionic run configuration

PhoneGap/Cordova/Ionic applications are executed according to a dedicated run/debug configuration.

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose PhoneGap/Cordova on the context menu.
2. In the Run/Debug Configuration: PhoneGap/Cordova dialog box that opens, specify the following:
   1. The name of the configuration.
   2. In the PhoneGap/Cordova Executable Path field, specify the location of the executable file `phonegap.cmd`, `cordova.cmd`, or `ionic.cml` (see Installing PhoneGap/Cordova/Ionic).
   3. In the PhoneGap/Cordova Working Directory field, specify the folder under which the PhoneGap/Cordova/Ionic application files to run are stored.
   4. From the Command drop-down list, choose the command to run. The contents of the drop-down list, depend on the actually used framework, namely, on the executable file specified in the PhoneGap/Cordova Executable Path field. The available options are:
      - For PhoneGap:
        - emulate
        - run
        - prepare
        - serve
        - remote build
        - remote run

        See https://www.npmjs.org/package/phonegap for a list of PhoneGap-specific commands with descriptions.
      - For Cordova:
        - emulate
        - run
        - prepare
        - serve

        See https://www.npmjs.org/package/cordova for a list of Cordova-specific commands with descriptions.
      - For Ionic:
        - emulate
        - run
        - prepare
        - serve

        See https://www.npmjs.org/package/ionic for a list of Ionic-specific commands with descriptions.
   5. From the Platform drop-down list, choose the platform for running on which the application is intended. The available options are:
      - Android
      - ios To emulate this platform, you need to install the ios-sim command line tool globally. You can do it through the Node Package Manager (npm), see Installing and Removing External Software Using Node Package Manager or by running the `sudo npm install ios-sim -g` command, depending on your oeprating system.
      - amazon-fireos
      - firefoxos
      - blackberry10
      - ubuntu
      - wp8
      - windows8
      - browser

      Learn more about targeted platforms at http://docs.phonegap.com/en/edge/guide_platforms_index.md.html#Platform%20Guides and http://cordova.apache.org/docs/en/4.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface .
   6. For Cordova and Ionic, specify the targeted virtual or physical Android device to run the application on: select the Specify Target check box and select the required

device from the drop-down list. The list shows all the virtual and physical devices that are currently configured on our machine. See http://docs.phonegap.com/en/edge/guide_platforms_android_index.md.html#Android%20Platform%20Guide for details.

If PhpStorm displays the following error message: Cannot detect ios-sim in path, make sure you have installed the `ios-sim`, see Before you start.

3. To run a PhoneGap/Cordova/Ionic application, select the required run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button ▶.

Markup Languages and Style Sheets

In this section:

- Markup Languages and Style Sheets
  - Markup and template languages support
  - Parsing Web contents
- Changing Color Values in Style Sheets
- Editing HTML Files
- Emmet Support
- Generating DTD
- Generating XML Schema from Instance Document
- Generating Instance Document from XML Schema
- Markdown
- Minifying CSS
- Navigating Between Edit Points
- Referencing XML Schemas and DTDs
- Transpiling Sass, Less, and SCSS to CSS
- Transpiling Stylus to CSS
- Using Handlebars and Mustache Templates
- Validating Web Content Files
- Viewing Styles Applied to a Tag
- Viewing Images
- Working with Sass and SCSS in Compass Projects

Note In this part you will find information that is specific for the web content files only!

Markup and template languages support

PhpStorm supports editing of files in the following markup and template languages:

- XML
- HTML/XHTML
- CSS
- Sass, SCSS
- Less
- Jade
- YAML
- Stylus
- Compass
- Handlebars expressions and Mustache templates

The markup languages and style sheets are integrated into PhpStorm and can use the most powerful editing features:

- Validation and syntax highlighting.
- Code completion (`Ctrl+Space`).
- Indentation (`Ctrl+Alt+I` , `Ctrl+Alt+L`).
- Formatting (`Ctrl+Alt+L`) according to the code style .
- Intention actions (`Alt+Enter`).
- Viewing code structure (`Alt+7`).
- Navigation in the source code (`Ctrl+B`).
- Integrated documentation (`Ctrl+Q`).
- Search for usages (`Alt+F7`).
- Commenting and uncommenting lines (`Ctrl+Slash` , `Ctrl+Shift+Slash`).
- Unwrapping and removing tags (`Ctrl+Shift+Delete`).

All these features work if PhpStorm successfully locates the DTD or schema file. In this case, all the files are validated against the DTD or schema, and the editing conveniences become available. Without a DTD or schema, only the well-formedness check is possible.

These features for web contents work same way as for the other source files. Refer to the respective topics of the Advanced Editing Features part for the detailed descriptions of procedures, and to Keyboard shortcuts .

Parsing Web contents

PhpStorm parses Web contents files according to the following specifications:

- HTML: specification is configurable in the Default HTML language level in the Schemas and DTDs page of the Settings/Preferences dialog. By default, specification HTML 5.0 from W3C is assumed.
- CSS: specification CSS 3.0. The most common selectors are supported: universal selector `*` , type selectors `.a` , descendant selectors `.a.b` , child selectors `.a .b` , ID selectors `#b` , pseudo-classes and class selectors `DIV.warning` .
- PhpStorm uses Xerces 2.11, an XML parser developed by Apache Software Foundation Group.

Changing Color Values in Style Sheets

On this page:

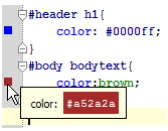
- Introduction
- Choosing color value
- Changing color value

Introduction

In addition to the editing techniques that are common for all file types, PhpStorm provides specific techniques for CSS, Sass, SCSS and Less files.

In particular, it is possible to easily change color values without the necessity to memorize and type color codes.

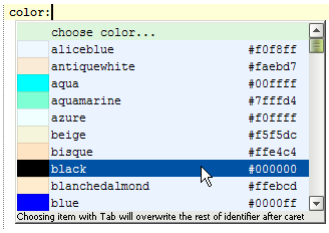
The `color` properties are marked with the icons of the corresponding color in the left gutter area of the editor. Use these icons to view colors and change color values.
When you point with your mouse cursor to the color icon, the pop-up window appears, showing the color and its code.



Choosing color value

To choose color value in a style sheet

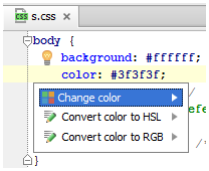
1. Open the desired style sheet for editing.
2. Type `color:` , and then press `Ctrl+Space` .
3. Select the desired color value from the suggestion list, or choose color... to pick a custom one:



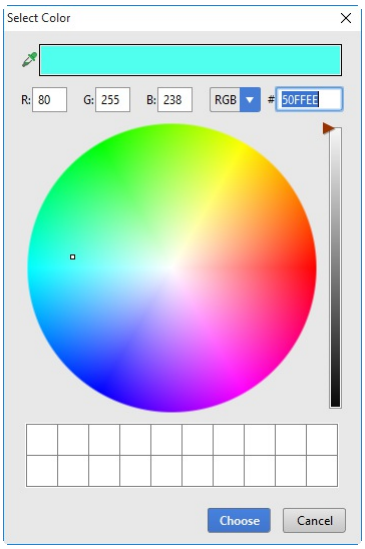
Changing color value

To change color value in a style sheet

1. Open the desired style sheet for editing, and locate color property to be changed.
2. Do one of the following:
   - If the color icon is shown , just double-click is in the left gutter of the editor.
   - If the color icon is not shown , then press `Alt+Enter` , or click to reveal the list of intention actions, and choose Change color intention action:



3. In the Choose color dialog box, pick the desired new color, and click Choose:

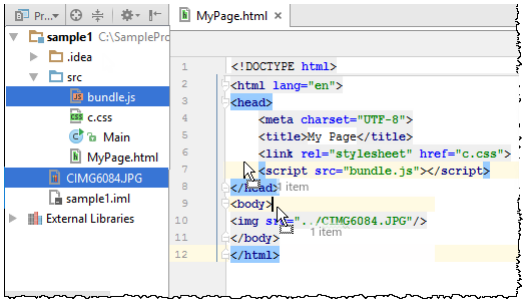


Editing HTML Files

PhpStorm enables you to create HTML files , and edit them.

When a file is opened for editing, you can drag a JavaScript or CSS file into the opened HTML file. PhpStorm will automatically create a script or a link tag inside the `<head>` tag.

Same way, you can drag an image to add an `img` tag with the `width` and `height` attributes anywhere inside the `<body>` tag.



Emmet Support

With PhpStorm, you can edit HTML and CSS code faster by applying Emmet features. Just type an Emmet abbreviation in HTML and press Tab to expand it into the markup. Emmet also works in the CSS and JSX context.

PhpStorm provides two types of Emmet support.

- Native
- Support of additional templates

In this section:

- Enabling Emmet Support
- Enabling Support of Additional Live Templates
- Configuring Abbreviation Expansion Key
- Expanding Emmet Templates with User Defined Templates
- Surrounding a Code Block with an Emmet Template

Enabling Emmet Support

On this page:

- Basics
- Enabling and configuring native Emmet support in the HTML or XML context
- Enabling native Emmet support in the JavaScript context
- Enabling and configuring native Emmet support in the CSS context

Basics

Native Emmet support allows you to generate XML/HTML, JavaScript (JSX Harmony) and CSS structures based on abbreviations . PhpStorm supports basic Emmet and Emmet version 1.1 features, such as:

- New syntax for writing RGBA colors.
- Implied attributes.
- Default attributes.
- Boolean attributes.
- The Update Tag action.

Emmet is supported in HTML/XML, JavaScript (JSX Harmony) and in the CSS contexts. This support is configured separately on the HTML , JSX and CSS pages respectively.

Enabling and configuring native Emmet support in the HTML or XML context

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click XML under Emmet. The Emmet page opens.
2. To enable the Emmet support in the HTML or XML context, select the Enable XML/HTML Emmet check box.
   When this check boxes is cleared, all the other controls on this page become disabled.
3. To have PhpStorm show a pop-up window with a preview of the entered abbreviation before actually expanding it , select the Enable abbreviation preview check box.
4. Specify how Emmet in PhpStorm will treat URL addresses by selecting or clearing the Enable automatic URL recognition while wrapping text with <a> tag check box.
   - If this check box is cleared and you attempt to wrap an URL address with the <a> tag, PhpStorm simply encloses the URL address in `<a href=""></a>` and positions the cursor inside the double quotes in the `href` attribute. For example, wrapping `http://www.jetbrains.com` will result in `<a href="` | `">http://www.jetbrains.com</a>` :

     ```
     <a href="">http://www.jetbrains.com</a>
     ```

   - If this check box is selected and you attempt to wrap an URL address with the <a> tag, PhpStorm inserts the URL address inside the double quotes as the value of the `href` attribute and encloses the URL in `<a href="<wrapped URL>"></a>` . For example, wrapping `http://www.jetbrains.com` will result in `<a href="http://www.jetbrains.com">http://www.jetbrains.com</a>` . Moreover, PhpStorm highlights the wrapped URL green as a recognized URL:

     ```
     <a href="http://www.jetbrains.com">http://www.jetbrains.com</a>
     ```

Enabling native Emmet support in the JavaScript context

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X.
2. Under the Languages and Frameworks node, click JavaScript , and select the language level JSX Harmony.
3. Expand the Editor node, and then click JSX under Emmet. The JSX page opens.
4. To enable the Emmet support in the JavaScript context, select the Enable JSX Emmet check box.

Enabling and configuring native Emmet support in the CSS context

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Editor node, and then click CSS under Emmet. The CSS page opens.
2. To enable the Emmet support in the CSS context, select the Enable CSS Emmet check box.
   When this check boxes is cleared, all the other controls on this page become disabled.
3. Configure the way unknown abbreviations are treated by selecting or clearing the Enable fuzzy search among CSS abbreviations check box: When this check box is selected, every unknown abbreviation will be scored against available template names. The match with the best score will be used to resolve the template. For example, with this option enabled, the following abbreviations can be equal to:
   - `ov:h`
   - `ov-h`
   - `o-h`
   - `oh`
4. Configure the way unrecognized properties are treated by selecting or clearing the Enable expansion of unknown properties ('unknown' to 'unknown:;') check box:
   - When this check box is selected, any entered word will be expanded into the same word followed with a colon and a semicolon;
   - When this check box is cleared, only known properties (for example, `color`) will be expanded this way (`color:;`)
5. Configure inserting browser-specific prefixes using the Auto insert CSS vendor prefixes check box: If this check box is selected, the CSS properties listed in the table below are expanded into constructs that contain pre-pending vendor prefixes. Learn more at Vendor prefixes .
   If this check box is cleared, the entire table of properties is disabled.
6. Configure the use of properties in different browsers using the Properties and vendor prefixes table. The table contains a list of CSS properties and vendor prefixes that correspond to various browsers.
   - To enable or disable a property in a browser, select or clear the check box under the browser column.
   - To add a new property to the list, click the Add button + or press `Alt+Insert` . Then type the name of the property in the dialog box that opens and enable it in the relevant browsers.
   - To delete one or more properties from the lsit, select them and press Remove — or press `Alt+Delete` .

Enabling Support of Additional Live Templates

Support of additional templates includes more than 200 different HTML, CSS, and XSL live templates. All of them are listed under the Emmet nodes on the Live Templates page of the Settings dialog box.

To enable support of additional HTML, CSS, and XSL live templates

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor.

2. On the Live Templates page, that opens, select the check boxes next to the relevant template groups.

Configuring Abbreviation Expansion Key

Shortcut key for expanding Emmet selectors and live templates is configurable. You can re-define this default setting for each specific live template.

Configuring a shortcut to expand abbreviations

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Emmet under Editor.

2. On the Emmet page that opens, choose the desired option from the Expand abbreviation with drop-down list.

Configuring a shortcut to expand a live template with

1. To re-define the expansion key for a live template, open the Live Templates page, expand one of the Zen Coding nodes, and select the desired template. The focus moves to the Template Text area.

2. From the Expand with drop-down list, select the key to expand the template with.
   This setting does not override the default setting specified for native Emmet support; you will just get the ability to expand the template using either of the specified keys.

Expanding Emmet Templates with User Defined Templates

You can expand Emmet templates with your own live templates.

Suppose you have a template entry with the following template text:

```
<entry type="$TYPES$">$END$ <entry>
```

To generate a list of entries, you just need to type `“entry-list<entry[number=$]*5”` and press `TAB`. By default, the `number` attribute will be generated before `type`. To customize the position where it is generated, you need to add the `ATTRS` variable to your template, for example:

```
<entry type="$TYPES$"
                    $ATTRS$>$END$ <entry>
```

The `ATTRS` variable must have an empty string as the default value and should be skipped.

To expand an Emmet template with a user-defined template

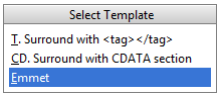
1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Live Templates under Editor.
2. On the Live Templates page that opens, expand the required Zen Coding template group, for example, `Zen HTML`, Zen CSS, or Zen XSL.
3. Select the template to expand. The focus moves to the Template Text area where the fields show the settings of the selected template.
4. In the Template Text field, add the required text and variables to the template body.
5. Click the Edit Variables button. In the Edit Template Variables dialog box that opens, specify the default variable values in the Default value field and select the Skip if defined check box, where necessary.

To surround a block of code with an Emmet template, follow these steps

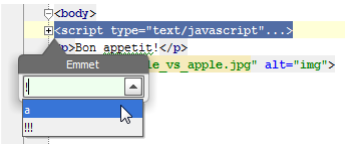
1. Open the desired file for editing and select a block of code to be surrounded.

2. Press `Ctrl+Alt+J`, or choose Code | Surround with Live Template on the main menu .

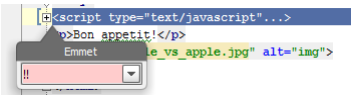
3. In the Select Template pop-up menu, choose Emmet:



4. Type the desired Emmet abbreviation, and press `Enter`.
   Note the drop-down list to the right. Clicking the down arrow reveals a history list of the recently applied Emmet live templates:



   Also mind the color indication. If you type a valid Emmet abbreviation, the background is green. However, when a non-existent abbreviation is entered, the background becomes red:



Generating DTD

A Data Type Definition (DTD) is required for running structure validation checks on a Web content file. PhpStorm can scan any XML file for the existing elements and attributes and generate a DTD for it.

To generate a DTD for an XML file

1. Open the desired file in the active editor tab.
2. On the main menu, choose Tools | XML Actions | Generate DTD From XML File. The resulting DTD is added as a section above the first line of the file.

Generating XML Schema From Instance Document

An XSD (XML Schema Definition) Schema is required for running structure validation checks on a Web content file. PhpStorm can scan any XML file for the existing elements and attributes and generate a Schema for it.

To have a Schema generated based on an XML instance document

1. With the desired XML document opened in the active editor tab, choose Tools | XML Actions | Generate XSD Schema from XML File on the main menu.

2. In the Generate Schema From Instance Document dialog box that opens configure the Schema generation procedure:
   - In the Instance Document Path text box, specify the location of the file to be used as the basis for Schema generation. By default, the field shows the full path to the current file. Accept this suggestion or click the Browse button and select the desired file in the dialog that opens .
   - In the Result Schema File Name text box, specify the name of the output file to place the generated Schema in.

     Warning! PhpStorm suggests the name of the source XML document with the `.xsd` extension. If you type another name, make sure the extension is correct.

   - Specify the location of the generated Schema. By default, the generated Schema file will be placed in the same directory as the source XML instance document. To specify another location, click the Browse button and select the desired path in the dialog that opens .
   - From the Design Type drop-down list, select the way to declare elements and complex types.
   - From the Detect Simple Content Type drop-down list, select the type to use for leaf text.
   - In the Detect Enumerations Limit text box, type the number of occurrences to cause the Schema enumeration appearance.

     Tip To suppress Schema enumeration, specify 0.

To generate an XML instance document from an XML Schema

1. With the desired Schema (`.xsd`) file opened in the active editor tab, choose Tools | XML Actions | Generate XML Document from XSD Schema on the main menu.

2. In the Generate Instance Document from Schema dialog box that opens configure the XML instance document generation procedure:
   - In the Schema Path text box, specify the location of the Schema to base the XML document generation on. By default, the field shows the full path to the current file. Accept this suggestion or click the Browse button ... and select the desired file in the dialog that opens .
   - In the Instance Document Name text box, specify the name of the output file to place the generated XML in.

     Warning! PhpStorm suggests the name of the source XML document with the `.xml` extension. If you type another name, make sure the extension is correct.

   - Specify the location of the generated document. By default, it will be placed in the same directory as the source Schema file. To specify another location, click the Browse button ... and select the desired path in the dialog that opens .
   - From the Element Name drop-down list, select the local name of the global element to be used as the root of the generated XML document.
   - Specify whether to take restriction and uniqueness particles into consideration by selecting the corresponding check boxes.

Markdown

On this page:

- Introduction
- Prerequisites
- Creating a Markdown file
- Markdown editor

Introduction

PhpStorm makes it possible to work with the Markdown files. PhpStorm recognizes syntax and provides initial support.

The Markdown files are marked with icon.

Prerequisites

Before you start working with Markdown, make sure that the Markdown Support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Markdown Support plugin is bundled with PhpStorm since version 2016.2.

Creating a Markdown file

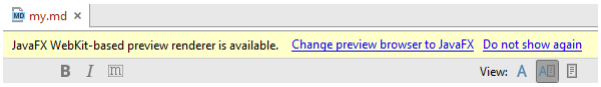
To create a Markdown file, follow these steps

1. Do one of the following:
   - Choose File | New on the main menu.
   - Right-click the target directory where the new file should be created, and choose New on the context menu.
   - Press `Alt+Insert`
2. Choose File.
3. In the New File dialog box, specify the new file name and extension `.md` .

The new file `<name>.md` , marked with icon, is created and opens for editing.

Markdown editor

The editor of a `.md` file by default shows the following:



Click one of the links to get rid of the banner. It's recommended to choose the link Change preview browser to JavaFX.

The editor is divided into two panes: the editor itself and the preview. Each of the panes can be hidden.

Editor pane		
Icon	Action	Description
B	Toggle bold mode	Inserts two asterisks before and after the selected text to render bold font.
I	Toggle italic mode	Inserts underscores before and after the selected text to render italic font.
m	Toggle monospaced (code span) mode	Inserts single apostrophes before and after the selected text to render monospaced font.
Preview pane		
A	Show editor only	Shows editor only with Markdown syntax.
A	Show editor and preview	Shows editor with Markdown syntax and the corresponding preview. The results of editing are immediately reflected in the preview pane.
	Show preview only	Shows preview that renders the Markdown syntax. Editing is not possible, and the buttons B , I and m are disabled.

Minifying CSS

On this page:

- Introduction
- Installing and configuring the YUI Compressor
- Creating a file watcher
- Minifying the code

Introduction

The term minification or compression in the context of CSS means removing all unnecessary characters, such as spaces, new lines, comments without affecting the functionality of the source code.

These characters facilitate working with the code at the development and debugging stage by improving the code readability. However at the production stage these characters become extraneous: being insignificant for code execution, they increase the size of code to be transferred. Therefore it is considered good practice to remove them before deployment.

PhpStorm supports integration with the YUI Compressor CSS minification tool.

In PhpStorm, minifier configurations are called File Watchers. For each supported minifier, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a minifier against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the minifier to use on your machine.

Installing and configuring the YUI Compressor

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The CSS minifier is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the CSS minifier.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the CSS minifier and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Download and install the JavaScript minification tool. The easiest way is to use the Node Package Manager (npm), which is a part of Node.js .
   1. Switch to the directory where the Node Package Manager (npm) is stored or define a `path` variable for it so it is available from any folder.
   2. Type the following command at the command line prompt:

```
npm install yuicompressor
```

If you use the Node Package Manager (npm), the minifier is installed under Node.js so Node.js, which is required for starting the tool, will be specified in the path to it.

Creating a file watcher

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
2. Click the Add button + or press `Alt+Insert` and choose the YUI Compressor CSS predefined template from the pop-up list.
3. In the Program text box, specify the path to the `yuicompressor-<version>.jar` file. If you installed the tool through the Node Package Manager, PhpStorm locates the required file itself and fills in the field automatically. Otherwise, type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.
4. Proceed as described on page Using File Watchers .

Minifying the code

When a Minification File Watcher is enabled (see Enabling and disabling File Watchers), minification starts automatically as soon as a file to which transpilation is applicable is changed or saved, see Configuring the behaviour of the File Watcher .

PhpStorm creates a separate file with the generated output. The file has the name of the source CSS file and the extension `min.css` . The location of the generated file is defined in the Output paths to refresh text box of the New Watcher dialog . However, in the Project Tree, it is shown under the source CSS file which is now displayed as a node.

Navigating Between Edit Points

In the HTML and XML context, you can navigate between edit points, that is, between important points of code where editing is possible.

- To move the cursor to the previous edit point, choose Navigate | Previous Emmet Edit Point, or press `Shift+Alt+Open Bracket`.
- To move the cursor to the next edit point, choose Navigate | Next Emmet Edit Point, or press `Shift+Alt+Close Bracket`.

Referencing XML Schemas and DTDs

Your XML file may reference an external XML schema (XSD) or DTD file, e.g.

```
<root   xmlns="http://www.example.org"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.example.org
                    http://www.example.org/xsds/example.xsd">
```

or

```
<!DOCTYPE root SYSTEM "http://www.example.org/dtds/example.dtd">
```

If the referenced URL or the namespace URI is "unfamiliar", it's marked as an error.

For situations like this, PhpStorm provides the following intention actions :

- Fetch External Resource. PhpStorm downloads the referenced file and associates it with the URL (or the namespace URI). The error highlighting disappears. The XML file is validated according to the downloaded schema or DTD. (The associations of the URLs and the namespace URIs with the schema and DTD files are shown on the Schemas and DTDs page in the Settings dialog .)
- Manually Setup External Resource. Use this option when you already have an appropriate schema or DTD file available locally. The Map External Resource dialog will open and you'll be able to select the file for the specified URL or namespace URI. The result of the operation is the same as in the case of fetching the resource.
- Ignore External Resource. The URL or the namespace URI is added to the Ignored Schemas and DTDs list. (This list is shown on the Schemas and DTDs page in the Settings dialog .) The error highlighting disappears. PhpStorm won't validate the XML file, however, it will check if the XML file is well-formed.

There is one more intention action that you may find useful: Add Xsi Schema Location for External Resource. This intention action lets you complete your root XML elements. If the namespace is already specified, PhpStorm can add a couple of missing attributes.

For example, if you have a fragment like this:

```
<root   xmlns="http://www.example.org">
```

and perform the intention action on the value of the `xmlns` attribute, the result will be:

```
<root   xmlns="http://www.example.org"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.example.org >
```

At this step, you'll be able to add the schema URL, and then map the URL (or the namespace URI) onto an appropriate schema file, or add the URL (or the URI) to the Ignored Schemas and DTDs list.

Transpiling Sass, Less, and SCSS to CSS

On this page:

- Introduction
- Installing the Sass/SCSS transpiler
- Installing the Less transpiler
  - Installing the Less transpiler globally
  - Installing the Less transpiler in a project
- Creating a file watcher
- Transpiling the code

Introduction

Sass, Less, and SCSS code is not processed by browsers that work with CSS code. Therefore to be executed, Sass, Less, or SCSS code has to be translated into CSS. This operation is referred to as transpilation and the tools that perform it are called transpilers.

PhpStorm supports integration with transpiler tools that translate Sass, Less, and SCSS code into CSS.

In PhpStorm, transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

Warning! Make sure the NodeJS plugin is enabled. You need to download it form the JetBrains plugin repository and activate it on the Plugins page of the Settings/Preferences dialog box as described in Enabling and Disabling Plugins.

Installing the Sass/SCSS transpiler

Sass and SCSS transpilers are managed through the Ruby Gem manager.

1. Download and install Ruby .
2. Specify a `path` variable for the folder where the Ruby executable file and the `gem.bat` file are stored. This lets you launch Ruby and Gem Manager from any folder and ensures that Ruby is successfully launched during transpilation.
3. Type one of the following commands at the command line prompt:

```
gem install sass
```

or

```
gem install scss
```

The tool is installed to the folder where Ruby executable file and the `gem.bat` file are stored.

Installing the Less transpiler

The easiest way to install the Less transpiler is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Less transpiler executable file, choose one of the following methods:

- Install the transpiler globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the transpiler in a specific project and thus restrict its use to this project.
- Install the transpiler in a project as a development dependency .

In either installation mode, make sure that the parent folder of the Less transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

Installing the Less transpiler globally

Global installation makes a transpiler available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the transpiler is automatically added to the `PATH` variable, which enables you to launch the transpiler from any folder. To install the transpiler globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

```
npm install -g less
```

The `-g` key makes the transpiler run in the global mode. Because the installation is performed through NPM, the Less transpiler is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

For more details on the NPM operation modes, see npm documentation . For more information about installing the Less transpiler, see https://npmjs.org/package/less .

- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the

global and at the project level. Click +.

3. In the Available Packages dialog box that opens, select the required package to install.
4. Select the Options check box and type `-g` in the text box next to it.
5. Optionally specify the product version and click Install Package to start installation.

Installing the Less transpiler in a project

Installing a transpiler in a specific project restricts its use to this project. To run project installation, do one of the following:

– Run the installation from the command line:

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install less
```

– Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the required package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the Less transpiler in a `node_modules` folder. If your project already contains such folder, the Less transpiler is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the Less transpiler in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the Less transpiler is installed there.

In either case, make sure that the parent folder of the Less transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

Creating a file watcher

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.
2. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
3. Click the Add button + or press `Alt+Insert`. Depending on the tool you are going to use, choose the appropriate predefined template from the pop-up list:
   – Less
   – Sass
   – SCSS
4. In the Program text box, specify the path to the compiler executable file or archive depending on the chosen predefined template.
   – `lessc.cmd` for Less If you installed the tool through the Node Package Manager, PhpStorm locates the required file itself at `<node.js_home>/node_modules/bin/lessc.cmd` and fills in the field automatically. Otherwise, type the path manually or click the Browse button and choose the file location in the dialog box that opens.
   – `sass.bat` for Sass
   – `scss.bat` for SCSS

   If you installed the Sass and SCSS tools through Ruby, PhpStorm locates the required files itself at `<ruby_home>/bin/sass.bat` `<ruby_home>/bin/scss.bat` respectively and fills in the field automatically. Otherwise, type the path manually or click the Browse button and choose the file location in the dialog box that opens.
5. Proceed as described on page Using File Watchers .

Transpiling the code

When you open a TypeScript file, PhpStorm checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, PhpStorm displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, PhpStorm starts it automatically upon the event specified in the New Watcher dialog .

– If the Immediate file synchronization check box is selected, the File Watcher is invoked as soon as any changes are made to the source code.

– If the Immediate file synchronization check box is cleared, the File Watcher is started upon save (File | Save All, `Ctrl+S`) or when you move focus from PhpStorm (upon frame deactivation).

PhpStorm creates a separate file with the generated output. The file has the name of the source Sass, Less, or SCSS file and the extension `css`. The location of the generated files is defined in the Output paths to refresh text box of the New Watcher dialog . However, in the Project Tree, they are shown under the source file which is now displayed as a node.

Transpiling Stylus to CSS

On this page:

- Introduction
- Before you start
- Installing the Stylus transpiler globally
- Installing the Stylus transpiler in a project
- Creating a file watcher
- Transpiling the code

Introduction

Stylus code is not processed by browsers that work with CSS code. Therefore to be executed, Stylus code has to be translated into CSS. This operation is referred to as transpilation and the tools that perform it are called transpilers.

PhpStorm supports integration with a transpiler tool that translates Stylus code into CSS.

In PhpStorm, transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

The easiest way to install the Stylus transpiler is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Stylus transpiler executable file, choose one of the following methods:

- Install the transpiler globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the transpiler in a specific project and thus restrict its use to this project.
- Install the transpiler in a project as a development dependency .

In either installation mode, make sure that the parent folder of the Stylus transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

Before you start

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Stylus transpiler is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Stylus transpiler.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Stylus transpiler and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing the Stylus transpiler globally

Global installation makes a transpiler available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the transpiler is automatically added to the `PATH` variable, which enables you to launch the transpiler from any folder. To install the transpiler globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g stylus
     ```

     The `-g` key makes the transpiler run in the global mode. Because the installation is performed through NPM, the Stylus transpiler is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the Stylus transpiler, see https://npmjs.org/package/stylus .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing the Stylus transpiler in a project

Installing a transpiler in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with

Embedded Local Terminal for details).

2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install stylus
```

– Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the required package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the Stylus transpiler in a `node_modules` folder. If your project already contains such folder, the Stylus transpiler is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the Stylus transpiler in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the Stylus transpiler is installed there.

In either case, make sure that the parent folder of the Stylus transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

Creating a file watcher

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.
2. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
3. Click the Add button + or press `Alt+Insert` and choose the Stylus predefined template from the pop-up list.
4. In the Program text box, specify the path to the executable file:
   – `stylus` for OS X and Unix.
   – `stylus.bat` for Windows.

   Type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.
5. Proceed as described on page Using File Watchers .

Transpiling the code

When you open a Stylus file, PhpStorm checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, PhpStorm displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, PhpStorm starts it automatically upon the event specified in the New Watcher dialog .

– If the Immediate file synchronization check box is selected, the File Watcher is invoked as soon as any changes are made to the source code.
– If the Immediate file synchronization check box is cleared, the File Watcher is started upon save (File | Save All, `Ctrl+S`) or when you move focus from PhpStorm (upon frame deactivation).

PhpStorm creates a separate file with the generated output. The file has the name of the source Stylus file and the extension `css` . The location of the generated files is defined in the Output paths to refresh text box of the New Watcher dialog . However, in the Project Tree, they are shown under the source file which is now displayed as a node.

Using Handlebars and Mustache Templates

On this page:

- Basics
- Preparing to use Handlebars expressions and Mustache templates
- Configuring processing of Handlebars expressions and Mustache templates

Basics

With PhpStorm, you can use Handlebars expressions and Mustache templates in dedicated Handlebars and Mustache files that have the extension `.hbs` or `.mustache` respectively. PhpStorm distinguishes these file types and processes their contents according to default or custom settings specified on the Templates page of the Settings/Preferences dialog.

For more details about Handlebars expressions and Mustache templates see http://handlebarsjs.com/ .

Preparing to use Handlebars expressions and Mustache templates

To prepare using Handlebars expressions and Mustache templates

1. To enable the Handlebars/Mustache support in PhpStorm, install and activate the Handlebars/Mustache repository plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

2. Add the Handlebars and Mustache file types to PhpStorm:
   1. Open the Settings dialog by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Then select File Types. Find more on page Accessing Settings .
   2. On the File Types page that opens, click the Add button +.
   3. In the New File Type dialog box that opens, specify the name of the new type Handlebars/Mustache and optionally provide a description. When you click OK, PhpStorm returns to the File Types page.
   4. Select the newly registered file type and in the Registered Patterns area, specify the patterns that define the files of this type `*.hbs` and `*.mustache` . To add a pattern, click the Add button + and type the pattern in the Add Wildcard dialog box that opens.

   Find more on page Creating and Registering File Types .

3. Create a file of the type Handlebars or Mustache and place your code there. To create a file, select the parent folder and on the context menu of the selection choose New | Handlebars and specify the file name in the dialog box that opens. Find more on page Creating Files from Templates .

Configuring processing of Handlebars expressions and Mustache templates

To configure processing of Handlebars expressions and Mustache templates

The configuration settings you specify apply only to dedicated Handlebars and Mustache files that have the extension `.hbs` or `.mustache` respectively.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the JavaScript node under Languages&Frameworks, and then click Templates under JavaScript. The Templates page opens. Switch to the Handlebars/Mustache area.

2. Enable or disable tag completion.
   - To have PhpStorm automatically insert the second closing curly brace (`}`) of a Handlebars expression as soon as you type the first closing one, select the Automatically insert closing tag check box.
     PhpStorm also recognizes triple stashes (`{{{`) that prevent escaping values inside expressions. In this case, PhpStorm automatically inserts two closing curly braces as soon as you type the first closing one.
   - When this check box is cleared, you have to type the closing curly braces and triple stashes manually.

3. To have Handlebars expressions and Mustache templates automatically reformatted during code generation, refactoring, or reformatting (`Ctrl+Alt+L`), select the Enable formatting check box.
   If the check box is cleared, the original formatting of Handlebars expressions and Mustache templates is preserved.

4. From the Language for comments drop-down list, select the language to inherit the style for comments from. When you enter a line or block comment by pressing `Ctrl+Slash` or `Ctrl+Shift+Slash` , PhpStorm inserts the comment delimiters that are used in the chosen language, for example, `{{!----}}` for Handlebars, `/**/` for JavaScript, `<!---->` for HTML, etc.

Validating Web Content Files

On this page:

- Types of validity checks
- Choosing the default HTML language level
- Choosing the default schema to validate XML files
- Running full validation on an XML file

Types of validity checks

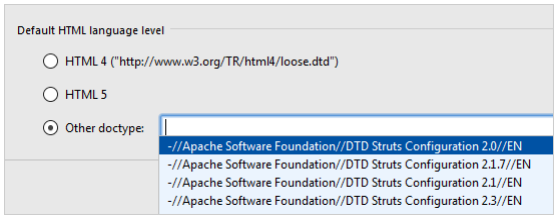
PhpStorm performs two different validity checks:

- On-the-fly validation is available for all Web content files and is performed as you edit the file. PhpStorm checks well-formedness, that is, detects various violations of syntax requirements, such as unclosed tags, wrong end-tag name, duplicate tags, unresolved links, etc. All encountered errors are highlighted in the editor.
  However, this form of code validation is rather *soft*, that is, not all requirements are taken into account.
- Full validation involves structure validation in addition to well-formedness check. Full validation is available for files that are associated with an XSD (XML Schema Definition) Schema or contain a Data Type Definition (DTD). PhpStorm checks whether the structure of your XML file complies with the structure defined in the corresponding DTD or Schema.
  The results of full validation are provided as a Message View.

Choosing the default HTML language level

Normally, an HTML or an XHTML file has the `<!DOCTYPE>` declaration which states the language level of the used in the source code from the file. This language level is used as a standard against which the contents of the file are validated. If an HTML or an XHTML file does not have a `<!DOCTYPE>` declaration, the contents of the file will be validated against the default standard (schema).

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Default XML Schemas under Schemas and DTDs.
   The Default XML Schemas page opens.
2. In the Default HTML Language Level area, choose the default schema to validate HTML and XHTML files without a `<!DOCTYPE>` declaration. The available options are:
   - HTML 4 or HTML 5: Choose one of these options to have files treated as HTML 4 or HTML 5 and validated against one of these standards.
   - Other doctype: Choose this option to have HTML files by default validated agains a custom DTD or schema and specify the URL of the DTD or schema to be used.
     Note that code completion is available in this field: press `Ctrl+Space` to see the list of suggested URLs.



   choose the XSD (XML Schema Definition) Schema to validate XML files. The available options are:
   - XML Schema 1.1 See W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures and W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes for details.
   - XML Schema 1.0 See XML Schema Part 1: Structures Second Edition and XML Schema Part 2: Datatypes Second Edition for details.

Choosing the default schema to validate XML files

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Default XML Schemas under Schemas and DTDs.
   The Default XML Schemas page opens.
2. In the Default XML Schema Version area, choose the XSD (XML Schema Definition) Schema to validate XML files. The available options are:
   - XML Schema 1.1 See W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures and W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes for details.
   - XML Schema 1.0 See XML Schema Part 1: Structures Second Edition and XML Schema Part 2: Datatypes Second Edition for details.

Running full validation on an XML file

1. Open the desired XML file in the editor, or just select it in the Project tool window.
2. On the context menu, choose Validate.

Viewing Styles Applied to a Tag

For HTML, XHTML, files, PhpStorm suggests a way to explore all styles applied to an arbitrary tag.

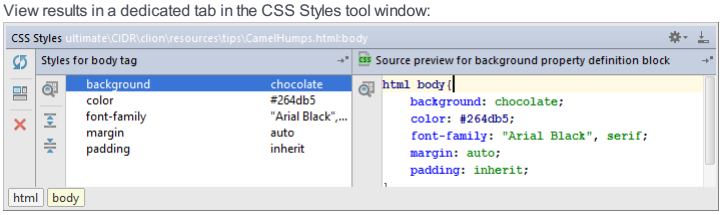
The results for each tag are displayed in the dedicated tabs of the CSS Styles tool window. Using this tool window, you can view the list of styles applied to a tag, and the definition of these styles. Besides that, you can navigate from style to the corresponding tag in the source code.

To show styles that are used for a tag

1. Open the desired file in the editor, and right-click the tag you want to explore for applied styles.

2. On the context menu, choose Show Applied Styles for Tag.

3. View results in a dedicated tab in the CSS Styles tool window:



Viewing Images

PhpStorm suggests several ways to view images, embedded in the HTML files. You can use navigation to source , open an image in an external graphical editor , or preview images on-the-fly . The viewing modes are configurable in the Images dialog .

To view an image in PhpStorm

1. In the Project tool window, select the desired image file.
2. On the context menu of the file, choose Jump to Source, or press `F4`.

Tip Configure the path to the desired external editor in the External Editor section of the Images dialog .

To view an image in an external editor

1. In the Project tool window, select an image file.
2. On the context menu of the file, choose Open in external editor, or press `Ctrl+Alt+F4`.

To preview an image, place the caret at a reference to an image in the editor, and do one of the following

- On the context menu, choose Jump to Source.
- Invoke the Navigate to declaration feature by pressing `Ctrl+B`.

Working with Sass and SCSS in Compass Projects

PhpStorm supports development in projects structured in compliance with the Compass framework . This framework uses Sass and SCSS extensions of CSS.

On this page:

- Preparing for Compass development
- Setting up a Compass project
- Integrating Compass with PhpStorm
- Creating a Compass Sass or a Compass SCSS transpiler
- Running a Compass Sass or Compass SCSS transpiler

Preparing for Compass development

The Compass framework is installed through the Ruby Gem manager, therefore you need to install Ruby first.

1. Download and install Ruby .
2. Specify a `path` variable for the folder where the Ruby executable file and the `gem.bat` file are stored. This lets you launch Ruby and Gem Manager from any folder and ensures that Ruby is successfully launched during transpilation.
3. Install and enable the Sass Support plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
4. Install Compass.
   1. The installation is performed in the command line mode. To start the built-in Terminal, hover your mouse pointer over in the lower left corner of the IDE, then choose Terminal from the menu (see Working with Embedded Local Terminal for details).
   2. Type the following command at the command line prompt:

      ```
      gem install compass
      ```

      The tool is installed to the folder where Ruby executable file and the `gem.bat` file are stored.

Setting up a Compass project

You can have a project set up according to the Compass requirements in two ways: create a new Compass project or create an empty project and introduce a Compass-specific structure in it. In either case, a project is set up through command line commands. Of course, you can set up a Compass project externally and then open it in PhpStorm.

During project set-up, a `conf.rb` configuration file is generated. You will need to specify the location of this file when integrating Compass with PhpStorm.

- To set up the Compass-specific structure in an existing project:
  1. Open the desired project in PhpStorm.
  2. Open the built-in Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu.
  3. At the command line prompt, type:

     ```
     compass init
     ```

- To create a Compass project from scratch:
  1. Open the desired project in PhpStorm.
  2. Open the built-in Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu.
  3. Switch to the folder that will be the parent for the new project folder. Type the following command:

     ```
     cd <parent folder name>
     ```

  4. At the command line prompt, type:

     ```
     compass create <the name of the project to be created>
     ```

Integrating Compass with PhpStorm

To develop a Compass-specific project in PhpStorm, you need to specify the Compass executable file `compass` and the project configuration file `config.rb`. You can do it either through the Compass Support page of the Settings dialog box or on the fly using an intention action that opens the Compass Support dialog box.

1. Open the Compass page or dialog box by doing one of the following:
   - Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Compass under Languages & Frameworks.
   - 1. In a `.sass` or `.scss` file, type the following import statement:

```
@import 'compass'
```

2. Click the red bulb icon or press `Alt+Enter`. Then choose Configure Compass from the suggestion list.

   The Compass Support dialog box opens.

2. To activate Compass support, select the Enable Compass support check box.
3. In the Compass executable file text box, specify the location of the `compass` executable file under the Ruby installation. Type the path manually, for example, `C:\Ruby200-x64\bin\compass`, or choose it from the drop-down list, or click the Browse button and choose the location of the `compass` file in the dialog box that opens.
4. In the Config path field, specify the location of the project Compass configuration file `config.rb`. Type the path manually, for example, `C:\my_projects\\compass_project\config.rb`, or choose it from the drop-down list, or click the Browse button and choose the location of the `compass` file int he dialog box that opens.
   The Compass configuration file `config.rb` is generated during project set-up through `compass create` or `compass init` commands.

Creating a Compass Sass or a Compass SCSS transpiler

Sass and SCSS are not processed by browsers that work with CSS code. Therefore to be executed, Sass or SCSS code has to be translated into CSS. This operation is referred to as transpilation and the tools that perform it are called transpilers.

PhpStorm supports integration with a transpiler tool that translates Sass and SCSS code from a Compass project without changing the Compass-specific project structure.

In PhpStorm, transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.
2. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
3. Click the Add button + or press `Alt+Insert` and choose the compass sass or compass scss predefined template from the pop-up list.
4. In the Program text box, specify the path to the executable file:
   - `compass.bat` for Windows
   - `compass` for Unix and OS X

   Type the path manually or click the Browse button and choose the file location in the dialog box that opens.
5. In the Arguments text box, type one of the following depending on the operating system used:
   - For OS X:
     - `compile $ProjectFileDir$` to process an entire directory
     - `compile $ProjectFileDir$ $FilePath$` to process a single file
   - For Windows:
     - `compile $UnixSeparators($ProjectFileDir$)$` to process an entire directory
     - `compile $UnixSeparators($FilePath$)$` to process a single file
   - For Linux (Ubuntu):
     - `compile $ProjectFileDir$` to process an entire directory
     - `compile $ProjectFileDir$ $FilePath$` to process a single file
6. Proceed as described on page Using File Watchers .

Running a Compass Sass or Compass SCSS transpiler

When you open a Sass or SCSS file, PhpStorm checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, PhpStorm displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, PhpStorm starts it automatically upon the event specified in the New Watcher dialog .

- If the Immediate file synchronization check box is selected, the File Watcher is invoked as soon as any changes are made to the source code.
- If the Immediate file synchronization check box is cleared, the File Watcher is started upon save (File | Save All, `Ctrl+S`) or when you move focus from PhpStorm (upon frame deactivation).

PhpStorm creates a separate file with the generated output. The file has the name of the source Sass or SCSS file and the extension `css`. The location of the generated files is defined in the Output paths to refresh text box of the New Watcher dialog . However, in the Project Tree, they are shown under the source file which is now displayed as a node.

Node.js

In this section:

- Node.js
  - Introduction
  - Before you start
  - Configuring Node.js in PhpStorm
- Configuring Node.js Interpreters
- Running and Debugging Node.js
- Running Nodeunit Tests
- V8 CPU and Memory Profiling
- Using Pug (Jade) Template Engine
- Running NPM Scripts
- Installing and Removing External Software Using Node Package Manager

Introduction

PhpStorm supports integration with the Node.js runtime environment thus enabling running, debugging, and unit testing of Node.js applications.

PhpStorm recognizes Node.js code and provides basic coding assistance and highlighting for it. To get guidance in Node development, see HowToNode.org .

Before you start

1. Download and install the Node.js runtime environment.
2. Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

The Node.js plugin introduces the following changes to the PhpStorm UI:

- Node.js page is added to the Settings dialog box.
- Run/debug configurations are added.

Configuring Node.js in PhpStorm

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks. The Node.js and NPM page opens.
2. In the Node Interpreter field, specify the local Node.js interpreter to use. Choose the interpreter from the drop-down list or click and choose the interpreter in the dialog box that opens.
   The term local Node.js interpreter denotes a Node.js installation on your computer. The term remote Node.js interpreter denotes a Node.js installation on a remote host or in a virtual environment set up in a Vagrant instance. On the Node.js and NPM page, you can specify only local interpreters. Remote interpreters are configured in the Configure Node.js Remote Interpreter Dialog dialog which can be accessed only from the Run/Debug Configuration: Node JS dialog. See Configuring Node.js Interpreters for details.
3. In the Code Assistance area, configure the Node.js core module sources if they are not configured yet.
   When developing a Node.js application it can be convenient to have code completion, reference resolution, validation, and debugging capabilities for Node core modules (`fs` , `path` , `http` , etc.). However, these modules are compiled into the Node.js binary. PhpStorm provides the ability to configure these sources as a JavaScript library and associate it with your project.
   - If the Node.js core module sources are not set up, PhpStorm displays a notification Node.js Core Library is not enabled with an Enable button. Click this button to have PhpStorm configure Node.js Core sources automatically.
     When the configuration is completed, PhpStorm displays information about the currently configured version, the notification Node.js Core Library is enabled, and adds two buttons: the Disable button and the Usage scope button.
   - If the library is set up, PhpStorm displays information about the currently configured version, the notification Node.js Core Library is enabled, and adds two buttons: the Disable button and the Usage scope button.
     - Click the Disable button to discard the configuration of the Node.js Core libraries in the current project.
     - Click the Usage scope button to associate the desired directories with libraries.
4. If necessary, configure the scope in which the Node.js Core sources are treated as libraries. Click the Usage scope button, and in the Usage Scope dialog box that opens, click the desired directories, and from the drop-down list select the newly configured Node.js core module sources library.
   The use of a library is enabled recursively, that is, if a library is associated with a folder it is automatically enabled in all the nested directories and files.

Configuring Node.js Interpreters

On this page:

- Introduction
- Configuring a local Node.js interpreter
- Configuring a remote Node.js interpreter on a host accessible through SSH connection
- Configuring a remote Node.js interpreter in a Vagrant environment instance
- Configuring a remote Node.js interpreter on a remote host accessible through SFTP
- Configuring mappings

Introduction

With PhpStorm, you can use local and remote Node.js interpreters.

The term local Node.js interpreter denotes a Node.js installation on your computer. The term remote Node.js interpreter denotes a Node.js installation on a remote host or in a virtual environment set up in a Vagrant instance.

You can access a remote interpreter in three ways:

- Using SSH credentials to access the host where the Node.js interpreter is installed.
- Through access to the corresponding Vagrant instance.
- According to a Server Access Configuration . This approach is also helpful if you are going to synchronize your project sources with the Web server on the target remote host.

Configuring a local Node.js interpreter

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, click the ... button next to the Node Interpreter drop-down list.
3. In the Node.js Interpreters Dialog that opens with a list of all the currently configured interpreters, click + in the toolbar. In the dialog box that opens, choose Add Local on the context menu , then choose the local installation of Node.js.
   The path to the chosen interpreter is shown in Node.js Interpreters Dialog .

Configuring a remote Node.js interpreter on a host accessible through SSH connection

Before you start:

1. Configure access to an `ssh` server on the target remote host and make sure this server is running.
2. Make sure the Node.js Remote Interpreter repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

To configure a Node.js interpreter using SSH credentials:

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose Node.js on the context menu. In the Run/Debug Configuration: Node JS dialog that opens, click ... next to the Node interpreter field.
2. In the Node.js Interpreters Dialog that opens with a list of all the currently configured interpreters, click + in the toolbar. In the dialog box that opens, choose Add Remote on the context menu .
3. In the Configure Node.js Remote Interpreter Dialog that opens, choose the SSH Credentials method.
4. Specify the name of the remote host and the port which the SSH server listens to. The default port number is 22.
5. Specify your credentials to access the remote host in accordance with the credentials received during the registration on the server. Type your user name and choose the authentication method:
   - To access the host through a password, choose Password from the Auth type drop-down list and type the password.
   - To access the host through a pair of SSH keys, choose Key pair, then specify the path to the file where your private key is stored or the passphrase if you have configured it during the generation of the key pair.
6. Specify the location of the Node.js executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/node` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the Node.js home directory must be open for edit.
7. When you click OK, PhpStorm checks whether the Node.js executable is actually stored in the specified folder.
   - If no Node.js executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the Node.js executable is found, you return to the Node.js Interpreters where the installation folder and the detected version of the Node.js interpreter are displayed.

Configuring a remote Node.js interpreter in a Vagrant environment instance

Before you start:

1. Make sure that Vagrant and Oracle's VirtualBox are downloaded, installed, and configured on your computer as described in Vagrant: Working with Reproducible Development Environments .
2. Make sure the Vagrant plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Make sure the Node.js Remote Interpreter repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
4. Make sure that the parent folders of the following executable files are added to the system PATH variable:
   - `vagrant.bat` or `vagrant` from your Vagrant installation. This should be done automatically by the Vagrant installer.
   - `VBoxManage.exe` or `VBoxManage` from your Oracle's VirtualBox installation.
5. Configure the Node.js development environment in the Vagrant instance to be used. Learn more about using Vagrant with PhpStorm in Vagrant: Working with Reproducible Development Environments .

To configure a Node.js Interpreter in a Vagrant instance

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose Node.js on the context menu. In the Run/Debug Configuration: Node JS dialog that opens, click ... next to the Node interpreter field.
2. In the Node.js Interpreters Dialog that opens with a list of all the currently configured interpreters, click + in the toolbar. In the dialog box that opens, choose Add Remote on the context menu .
3. In the Configure Node.js Remote Interpreter Dialog that opens, choose the Vagrant method.
4. Specify the Vagrant instance folder which points at the environment you are going to use. Technically, it is the folder where the VagrantFile configuration file for the desired environment is located. Based on this setting, PhpStorm detects the Vagrant host and shows it as a link in the Vagrant Host URL read-only field.
5. Specify the location of the Node.js executable file in accordance with the development environment configured in the selected Vagrant instance. By default PhpStorm suggests the `/usr/bin/node` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the Node.js home directory must be open for edit.
6. When you click OK, PhpStorm checks whether the Node.js executable is actually stored in the specified folder.
   - If no Node.js executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the Node.js executable is found, you return to the Node.js Interpreters where the installation folder and the detected version of the Node.js interpreter are displayed.

Configuring a remote Node.js interpreter on a remote host accessible through SFTP

Before you start:

1. Make sure a `ssh` server is running on the target remote host and you have configured access to it.
2. Make sure the Node.js Remote Interpreter and Remote Hosts Access plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
3. Make sure you have at least one PhpStorm-wide server access configuration of the type SFTP to establish access to the target host. To make a configuration available in all PhpStorm projects, clear the Visible only for this project check box in the Deployment: Connection Tab . See Creating a Remote Server Configuration for details.

To configure a remote Node.js interpreter based on an SFTP server access configuration

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose Node.js on the context menu. In the Run/Debug Configuration: Node JS dialog that opens, click ... next to the Node interpreter field.
2. In the Node.js Interpreters Dialog that opens with a list of all the currently configured interpreters, click + in the toolbar. In the dialog box that opens, choose Add Remote on the context menu .
3. In the Configure Node.js Remote Interpreter Dialog that opens, choose the Deployment Configuration method.
4. From the Deployment Configuration drop-down list, choose the server access configuration of the type SFTP according to which you want PhpStorm to connect to the target host. If the settings specified in the chosen configuration ensure successful connection, PhpStorm displays the URL address of the target host as a link in the Deployment Host URL field.
   To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . By default, PhpStorm retrieves path mappings from the chosen server access (deployment) configuration. If the configuration does not contain path mappings, PhpStorm displays the corresponding error message.

   To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the relevant server access configuration, switch to the Mappings tab, and map the local folders to the folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.
5. Specify the location of the Node.js executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/node` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the Node.js home directory must be open for edit.
6. When you click OK, PhpStorm checks whether the Node.js executable is actually stored in the specified folder.
   - If no Node.js executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway.
   - If the Node.js executable is found, you return to the Node.js Interpreters where the installation folder and the detected version of the Node.js interpreter are displayed.

Configuring mappings

When you debug an application with a remote Node.js interpreter, the debugger tells PhpStorm the name of the currently processed file and the number of the line to be processed. PhpStorm opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called mapping, it is set in the debug configuration.

If you use an interpreter accessible through SFTP connection or located on a Vagrant instance, the mappings are automatically retrieved from the corresponding deployment configuration or `Vagrantfile` . To specify additional mappings:

1. On the main menu, choose Run | Edit Configurations. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose Node.js on the context menu.
2. In the Run/Debug Configuration: Node JS dialog that opens, choose the required remote interpreter from the Node interpreter drop-down list.
3. Click ... next to the Path Mappings field.
4. The Edit Project Path Mappings Dialog that opens, shows the path mappings retrieved from the deployment configuration or `Vagrantfile` . These mappings are read-only.
   - To add a custom mapping, click + and specify the path in the project and the corresponding path on the remote runtime environment in the Local Path and Remote Path fields respectively. Type the paths manually or click ... and select the relevant files or folders in the dialog box that opens.
   - To remove a custom mapping, select it in the list and click —.

Running and Debugging Node.js

On this page:

- Before you start
- Local and remote modes of running or debugging Node.js applications
- Running a Node.js application
- Debugging a Node.js application locally
- Debugging a running Node.js application
- Creating a Node.js run/debug configuration
- Enabling Live Editing during a Node.js debugging session
- Creating a Node.js remote debug configuration.
- Node.js multiprocess debugging

Before you start

Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Local and remote modes of running or debugging Node.js applications

Running a Node.js application in PhpStorm is supported only in the local mode. This means that PhpStorm itself starts the Node.js engine and the target application according to a run configuration and gets full control over the session.

Debugging can be performed in two modes:

- Locally, with the Node.js engine started from PhpStorm.
- Remotely, when PhpStorm connects to an already running Node.js application. This approach gives you the possibility to re-start a debugging session without re-starting the Node.js server.

In either case, the debugging session is initiated through a debug configuration

You can also configure the behaviour of the browser and enable debugging the client-side code of the application. This functionality is provided through a `JavaScript Debug` run configuration, so technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated NodeJS run configuration.

Running a Node.js application

1. Create a Node.js run configuration .
2. To launch the application, select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button . The Run tool window opens.
3. Open the browser of your choice and open the page with the URL address generated through the server.listen function based on the `port` and `host` parameters. The page shows the result of executing your Node.js application.

Debugging a Node.js application locally

1. Set the breakpoints in the Node.js code, where necessary. At least one breakpoint is necessary otherwise the program will be just executed. If you want the debugging tool to stop at the first line of your code, set a breakpoint at the first line.
2. Create a Node.js run/debug configuration .
3. To start a debugging session, select the required debug configuration from the list on the main tool bar and then choose Run | Debug <configuration name> on the main menu or click the Debug toolbar button .
4. Open the browser of your choice and open the starting page of your application. Control over the debugging session returns to PhpStorm.
5. Switch to PhpStorm, where the controls of the Debug tool window are now enabled. Proceed with the debugging session step through the breakpoints , switch between frames, change values on-the-fly, examine a suspended program , evaluate expressions , and set watches .

Debugging a running Node.js application

With PhpStorm, you can connect to an already running Node.js applications. The application can be started either on the same machine or on a physically remote host.

When the application to debug is running on a physically remote host, you need to run a proxy or any other software that ensures port forwarding on the Node.js server. This is necessary because the debug port can open only on the `localhost` network interface. The `localhost` network interface cannot be accessed from another machine therefore PhpStorm cannot connect to it upon initiating a debugging session.

1. Make sure the application to debug has been launched in the target environment with the following parameters: `--debug-brk=<port through which the debugger on the remote host interacts with the network interface which accepts external connections>`
   Please note the following:
   - With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start.
   - With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later.
2. Create a Node.js Remote Debug configuration: in the Debug port text box, type the port number through which you will interact with the remote host according to the server access configuration, see Creating a Remote Server Configuration .
3. With the application still running, launch the Node.js Remote Debug configuration (select the configuration in the list and click the Debug toolbar button).
4. In the Run tool window, copy the URL address of the server and open the corresponding page in the browser. Control over the debugging session returns to PhpStorm.
5. Switch to PhpStorm. In the Debug tool window, step through the breakpoints , switch between frames, change values on-the-fly, examine a suspended program , evaluate expressions , and set watches .

Creating a Node.js run/debug configuration

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. In the Edit Configuration dialog box that opens, click the Add New Configuration toolbar button , and choose Node.js on the context menu.

3. In the Run/Debug Configuration: Node.js dialog box, that opens, specify the following:
   - The name of the configuration.
   - In the Node Interpreter field, specify the Node.js installation to use. Choose the local or remote interpreter from the drop-down list, or click ... and choose the interpreter in the dialog box that opens, or configure an interpreter as described in Configuring Node.js Interpreters .
   - To enable remote debugging of the application, specify the following option in the Node parameters text box: `--debug-brk=<port for connect to debugger remotely>`
     - With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start.
     - With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later.
   - In the JavaScript File field, specify the location of the file to start running the Node.js application from.
   - If the file to run references any other files, specify their location in the Working directory field.
   - If applicable, in the Application parameters text box, specify the arguments to be passed to the application on start through the process.argv array.
4. If necessary, configure the behaviour of the browser and enable debugging the client-side code of the application. This functionality is provided through a `JavaScript Debug` run configuration, so technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated NodeJS run configuration.
5. Click OK, when ready.

Enabling Live Editing during a Node.js debugging session

You can configure the behaviour of the browser and enable debugging the client-side code of the application. This functionality is provided through a `JavaScript Debug` run configuration, so technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated NodeJS run configuration.

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. From the list, choose the Node.js run configuration to activate the Live Edit functionality in. In the dialog box that opens, switch to the Browser / Live Edit tab.
3. Select the After launch check box to have a browser started automatically after a debugging session is launched. Specify the browser to use in the drop-down list next to the check box.
   - To use the system default browser, choose Default.
   - To use a custom browser, choose it from the list. Note that Live Edit is fully supported only in Chrome.
   - To configure browsers, click the Browse button ... and adjust the settings in the Web Browsers dialog box that opens. For more information, see Configuring Browsers .
4. Select the With JavaScript debugger check box to enable the JavaScript debugger in the selected browser and specify the URL address to open the application at.

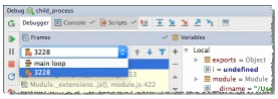
Creating a Node.js remote debug configuration.

1. On the main menu, choose Run | Edit Configurations.
2. In the Edit Configuration dialog box, that opens, click the Add New Configuration toolbar button +, and choose Node.js Remote Debug on the context menu.
3. In the Run/Debug Configuration: Node.js Remote Debug dialog box, that opens, specify the following:
   - The name of the configuration.
   - The host where the target application is running.
   - The port to connect to. Copy the port number from the information message in the Run tool window that controls the running application.
4. Click OK, when ready.

Node.js multiprocess debugging

PhpStorm supports debugging additional Node.js processes that are launched by the child_process.fork() method or by the cluster module . Such processes are shown as threads in the Frame pane on the Debugger tab of the Debug Tool Window .

1. Set the breakpoints in the processes to debug.
2. Create a Node.js run/debug configuration.
3. Choose the newly created configuration in the Select run/debug configuration drop-down list on the tool bar and click the Debug toolbar button .
   The Debug tool window opens and the Frames drop-down list shows the additional processes as threads as soon as they are launched:



To examine the data (variables, watches, etc.) for a process, select its thread in the list and view its data in the Variables and Debug Tool Window. Watches panes. When you select another process, the contents of the panes are updated accordingly.

Running Nodeunit Tests

On this page:

- Introduction
- Creating and running unit tests for Node.js applications
- Enabling unit testing for Node.js applications
- Creating Nodeunit tests
- Creating a Nodeunit run configuration

Introduction

PhpStorm supports integration with the nodeunit framework thus enabling running unit test for Node.js applications.

This topic provides guidelines in Node.js-specific unit testing procedures. For general information on testing in PhpStorm, see the section Testing .

Creating and running unit tests for Node.js applications

To create and run unit tests for a Node.js application, perform these general steps

1. Enable nodeunit support .
2. Write the unit tests .
3. To have PhpStorm recognize the unit tests and detect the corresponding production source code, mark the folder where the unit tests are stored as test folder .
4. Create a run configuration of the type Nodeunit.
5. Launch unit tests and monitor test results in the Run tool window.

Enabling unit testing for Node.js applications

To enable unit testing for Node.js applications

1. Download, install, and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the Node.js runtime environment.
3. Download and install the nodeunit testing framework

Creating Nodeunit tests

To create Nodeunit tests

- Create a folder `test` at the same level as the `src` folder
- Populate the `test` folder. For each production file, create a separate test file.
- Mark the folder where the tests are stored as test folder .

Creating a Nodeunit run configuration

To create a Nodeunit run configuration

1. Open the Run/Debug Configuration dialog box by choosing Run | Edit Configurations on the main menu.
2. Click the Add button + on the toolbar and select the Nodeunit configuration type.
3. In the dialog box that opens, specify the following:
   1. The name to identify the configuration.
   2. The path to the NodeJS installation to use.
      If you have appointed one of the installations as default , the field displays the path to its executable file.
   3. The working directory. This can be the project root folder or the parent directory for the `test` folder.
   4. The scope of tests to run.
      - To have PhpStorm run all the test files in a folder, choose All JavaScript test files in the directory from the Run drop-down list. In the Directory text box, type the path to the test folder relative to the working directory .
      - To have a specific test executed, choose JavaScript test file from the Run drop-down list. In the JavaScript test file text box, type the path to the file relative to the working directory .
4. Apply the changes and close the dialog box.

V8 CPU and Memory Profiling

On this page:

- Introduction
- Why is profiling important
- Before you start
- Preparing for V8 CPU and memory heap profiling
  - Installing the v8-profiler Package
- CPU profiling
  - Configuring CPU profiling
  - Collecting CPU profiling information
  - Analyzing CPU profiling information
    - Exploring call Trees
    - Analyzing the Flame chart
      - Selecting a Fragment in the Timeline
      - Synchronization in the Flame Chart
- Memory profiling
  - Configuring memory profiling
  - Collecting memory profiling information
  - Analyzing memory profiling information
    - Navigating through a snapshot

Introduction

As you may know, V8 is an open-source JavaScript engine developed by Google. It is used in Google Chrome, Chromium , Node.js, and io.js .

V8 has a sampling CPU profiler V8 profiler which is intended for capturing and analyzing CPU profiles and heap snapshots for your Node.js applications. With V8 CPU profiling you can get a better understanding of which parts of your code take up the most CPU time, and how your code is executed and optimized by the V8 JavaScript engine.

You can also open and explore profiles and snapshots captured in Google Chrome DevTools for your client-side code.

Why is profiling important

- A carefully designed algorithm can make your code faster and manage your memory consumption better, even more efficiently than the virtual machine can. Profiling is the way to look inside the execution of your code and prove your assumptions about your design decisions.
- Profiling is especially relevant for Javascript, which is a powerful language with advanced features like dynamic typing, closures, and even the ability to create code at runtime. Therefore the behavior of the JavaScript engine is quite sophisticated, and there are cases when you really need to go deep into the details of how the engine works. Some of the code patterns you use need being tweaked to allow the JavaScript optimizer to do its work.
- JavaScript is by no means a simple language to judge if your code manages memory well. Closure memory leaks are a good illustration of the fact that code that looks good and simple may still cause leaks.

Before you start

1. Install the Node.js runtime environment version 0.11.0 or higher.
2. Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Restart PhpStorm for the changes to take effect.

Preparing for V8 CPU and memory heap profiling

V8 CPU profiling is provided through the PhpStorm built-in functionality, you do not need to install any additional software.

V8 memory heap profiling is provided through the v8-profiler package.

The v8-profiler package can be installed from PhpStorm using the Node Package Manager (npm).

Installing the v8-profiler Package

You can install the v8-profiler package using the Node Package Manager (npm) either in the command line mode or on the Node.js and NPM page of the Settings dialog box.

To install the v8-profiler package globally, do one of the following:

- In the command line mode, type `npm install -g v8-profiler`.
- Use the Node.js and NPM page of the Settings dialog box:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, click Install + in the Packages area.
  3. In the Available Packages dialog box that opens, select the v8-profiler package from the list, possibly using the search field.
  4. Select the Options check box and type `-g` in the text box next to it to have the package installed globally and thus make it available from any PhpStorm project.
  5. Click Install.
  6. When the package is installed successfully, close the dialog box. PhpStorm brings you back to the Node.js and NPM page where the package is added to the list.

Learn more about installing packages through NPM at Installing and Removing External Software Using Node Package Manager .

CPU profiling

With V8 CPU profiling you can get a better understanding of which parts of your code take up the most CPU time, and how your code is executed and optimized by the V8 JavaScript engine.

To identify the processes that consume most of your CPU, you can use two methods: sampling and tracing.

- When the sampling method is applied, you periodically record stack traces of your application. The periods between records are measured in conventional units referred to as ticks.
  This method does not guarantee very good accuracy or precision for the following reason: snapshots are taken at random moments therefore any function can happen to be recorded in a snapshot. However, sampling can give us a rough picture of where the most of time is spent.
- When the tracing method is used, we actively record tracing information by ourselves, directly in the code. It is obviously better to get exact measurements of how much time each method took, and also allows to count how many times the traced method was called. The disadvantage of this method is that it comes with bigger result distortion compared to sampling.

Result Distortion. Both sampling and tracing introduce delays into execution and therefore influence the profiling results. With sampling, delays can be estimated as some fixed amount of time for each sampling event and do not introduce greater distortion than the sampling method itself (i.e. the delay is much shorter than the sampling interval). With tracing, the profiling delay depends on the code and the places where we made tracing measurements. For instance, if a traced method is called inside other traced methods numerously, all inner delays will accumulate for the outer method. If so, it may be difficult to separate the execution time from tracing distortion.

Usually we use sampling and tracing methods together. We start with sampling to get an idea of which parts of our code take the most time, and then instrument the code with tracing calls to zero on the issues.

Measurements are made not only for the work of your code, but also activities performed by the engine itself, such as compilation, calls of system libraries, optimization, and garbage collection. The following time metrics are made for execution of functions themselves and for performing activities:

- Total: the number of ticks (the time) during which a function was executed or an activity was performed.
- Total%: the ratio of a function/activity execution time to the entire time when measurements were made.
- Self: the pure execution time of a function/activity itself, without the time spent on executing functions called by it.
- Self%: the ratio of the pure execution time of a function/activity to the entire time when the measurements were made.
- Of Parent: the ratio of the pure execution time of a function to the execution time of the function that called it (Parent).

Configuring CPU profiling

To invoke V8 CPU profiling on application start, you need to specify additional settings in the Node.js run configuration according to which the application will be launched.

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. From the list, choose the Node.js run configuration to activate CPU Profiling in or create a new configuration as described in Running and Debugging Node.js .
3. Switch to the V8 Profiling pane and specify the following:
   1. Select the Record CPU profiling info check box.
   2. In the Log folder field, specify the folder to store recorded logs in. Profiling data are stored in V8 log files `isolate-<session number>` .

Collecting CPU profiling information

1. Select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button .
2. When the scenario that you need to profile is executed, stop the process by clicking on the tool bar of the Run tool window.

V8 log file will be processed by V8 scripts to calculate averaged call traces. PhpStorm opens the V8 Profiling Tool Window .

Analyzing CPU profiling information

Analyzing the profiling logs is available after the process stops because currently stopping and restarting profiling during execution of an application is not supported.

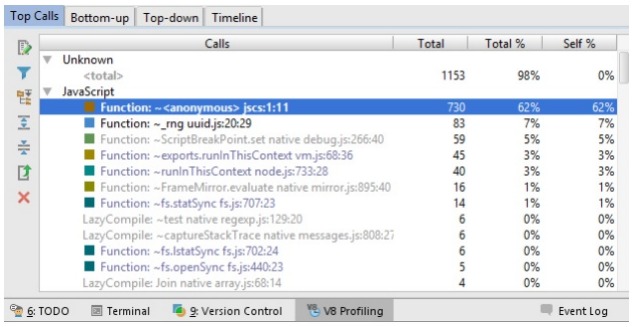
The collected profiling data is displayed in the V8 Profiling Tool Window which PhpStorm opens automatically when you stop your application. If the window is already opened and shows the profiling data for another session, a new tab is added. Tabs that were opened automatically are named after the run configurations that control execution of the applications and collecting the profiling data.

If you want to open and analyze some previously saved profiling data, choose V8 Profiling - Analyze V8 Profiling Log on the main menu and select the relevant V8 log file `isolate-<session number>` . PhpStorm creates a separate tab with the name of the log file.

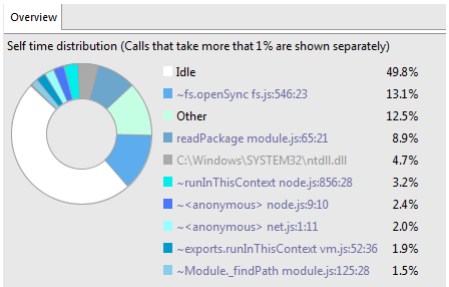
Exploring call Trees

Based on the collected profiling data, PhpStorm builds three call trees and displays each of them in a separate pane. Having several call trees provides the possibility to analyze the application execution from two different points of view: on the one hand, which calls were time consuming ("heavy"), and on the other hand, "who called whom".

- The Top Calls pane shows a list of performed activities sorted in the descending order by the Self metrics. For each activity PhpStorm displays its Total, Total%, and Self% metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined.

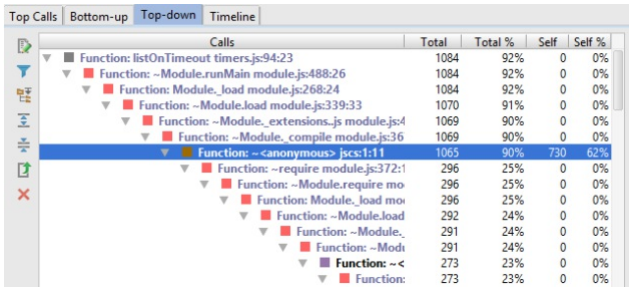


The diagram in the Overview pane shows distribution of self time for calls with the Self% metrics above 1%.



- The Bottom-up pane also shows the performed activities sorted in the descending order by the Self metrics. Unlike the Top Calls pane, the Bottom-up pane shows only the activities with the Total% metrics above 2 and the functions that called them. This is helpful if you encounter a heavy function and want to find out where it was called from.
  For each activity PhpStorm displays its execution time in ticks and the Of Parent metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined.

- The Top-down pane shows the entire call hierarchy with the functions that are execution entry points at the top. For each activity PhpStorm displays its Total, Total%, Self, and Self% metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined. Some of the functions may have been optimized by V8, see Optimizing for V8 for details.
  - The functions that have been optimized are marked with an asterisk (*) before the function name.
  - The functions that possibly require optimization but still have not been optimized are marked with a tilde (~) character before the function name. Though optimization may be delayed by the engine or skipped if the code is short-running, a tilde (~) points at a place where the code can be rewritten to achieve better performance.



- To navigate to the source code of a function, select the function in question in the tree and click on the toolbar or choose Jump to source on the context menu of the selection. The file with the source code of the selected function is opened in the editor with the cursor positioned at the function.

- When a tab for a profiling session is opened, by default the nodes with heaviest calls are expanded. While exploring the trees, you may like to fold some nodes or expand other ones. To restore the original tree presentation, click the Expand Heavy Traces button on the toolbar.

- To have PhpStorm display only the calls that indeed cause performance problems, filter out light calls:
  1. Click the Filter button on the toolbar.
  2. Using the slider, specify the minimum Total% or Parent% value for a call to be displayed and click Done.

- To expand or collapse all the nodes in the active pane, click or on the toolbar respectively.

- To expand or collapse a node, select it and choose Expand Node or Collapse Node on the context menu of the selection.

- Save and compare calls and lines:
  - To save a line with a function and its metrics, select the function and choose Copy on the context menu of the selection. This may be helpful if you want to compare the measurements for a function from two sessions, for example, after you make some improvements to the code.
  - To save only the function name and the name of the file where the function is defined, select the function and choose Copy Call on the context menu of the selection.
  - To compare an item with the contents of the Clipboard, select the item in question and choose Compare With Clipboard on the context menu of the selection. Compare the items in the Difference Viewer that opens.

- To save the call tree in the current pane to a text file, click on the toolbar and specify the target file in the dialog box that opens.

Analyzing the Flame chart

Use the multicolor chart in the Flame Chart tab to find where the application paused and explore the calls that provoked these pauses. The chart consists of four areas:

- The upper area shows a timeline with two sliders to limit the beginning and the end of a fragment to investigate.
- The bottom area shows a stack of calls in the form of a multicolor chart. When called for the first time, each function is assigned a random color, whereupon every call of

this function within the current session is shown in this color.

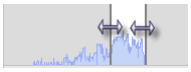
- The middle area shows a summary of calls from the Garbage Collector, the engine, the external calls, and the execution itself. The colors reserved for the Garbage Collector, the engine , the external calls, and the execution are listed on top of the area:

- The right-hand pane lists the calls within a selected fragment, for each call the list shows its duration, the name of the called function, and file where the function is defined.

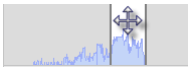
Selecting a Fragment in the Timeline

To explore the processes within a certain period of time, you need to select the fragment in question. You can do it in two ways:

- Use the sliders:



- Click the window between two sliders and drag it to the required fragment:



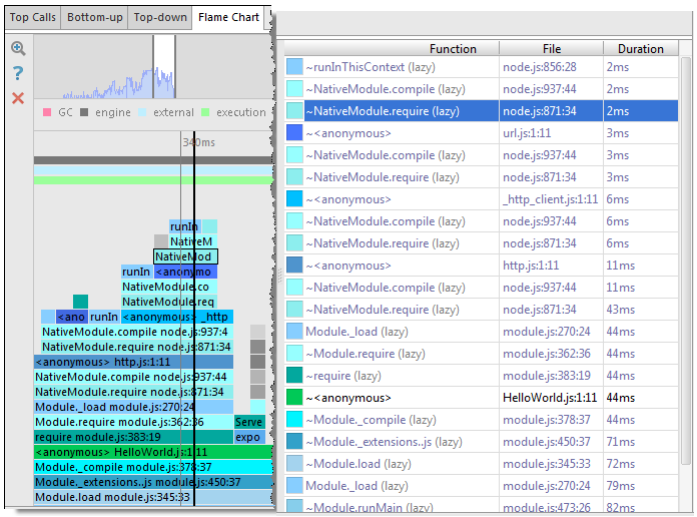
In either case, the multicolor chart below shows the stack of calls within the selected fragment.

To enlarge the chart, click the selected fragment and then click the Zoom button on the toolbar. PhpStorm opens a new tab and shows the selected fragment enlarged to fit the tab width so you can examine the fragment with more details.

Synchronization in the Flame Chart

The bottom and the right-hand areas are synchronized: as you drag the slider in the bottom area through the timeline the focus in the right-hand pane moves to the call that was performed at each moment.

Moreover, if you click a call in the bottom area, the slider moves to it automatically and the focus in the right-hand pane switches to the corresponding function, if necessary the list scrolls automatically. And vice versa, if you click an item in the list, PhpStorm selects the corresponding call in the bottom area and drags the slider to it automatically:



PhpStorm supports navigation from the right-hand area to the source code of called functions, to the other panes of the tool window, and to areas in the flame chart with specific metrics.

- To jump to the source code of a called function, select the call in question and choose Jump to Source on the context menu of the selection.
- To switch to another pane, select the call in question, choose Navigate To on the context menu of the selection, and then choose the destination:
  - Navigate in Top Calls
  - Navigate in Bottom-up
  - Navigate in Top-down

  PhpStorm switches to the selected pane and moves the focus to the call in question.
- To have the flame chart zoomed at the fragments with specific metrics of a call, select the call in question, choose Navigate To on the context menu of the slection, and then choose the metrics:
  - Navigate to Longest Time
  - Navigate to Typical Time
  - Navigate to Longest Self Time
  - Navigate to Typical Self Time

You can also navigate to the stacktrace of a call to view and anlyze exceptions. To do that, select the call in question and choose Show As Stacktrace. PhpStorm opens the stacktrace in a separate tab, to return to the Flame Chart pane, click V8 CPU Profiling tool window button in the bottom tool window.

Memory profiling

Though the V8 Javascript engine does memory management for you, memory leaks or dynamic memory problems are still possible. Here are some examples of memory leaks or shortcomings reasons:

- Using global objects to store collections of data, with complicated free policies.
- Errors in usages of closures: closures keep references onto outside objects.
- Keeping detached DOM nodes in javascript variables.
- Too frequent memory allocation.

Memory management control is especially important for Node.js applications, because the server-side code tend to run long while memory inaccuracy is accumulated.

V8 heap snapshots are complicated by their nature; they include many “engine” objects; the inner structure of objects differs from what you expect while reading your code. Learn more at Javascript Memory Profiling .

Configuring memory profiling

To allow taking memory snapshots, you need to specify additional settings in the Node.js run configuration according to which the application will be launched.

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. From the list, choose the Node.js run configuration to activate CPU Profiling in or create a new configuration as described in Running and Debugging Node.js .
3. Switch to the V8 Profiling pane and specify the following:
   1. Select the Allow taling heap snapshots check box.
   2. In the Log folder field, specify the folder to store recorded logs in. Profiling data are stored in V8 log files `isolate-<session number>` .
   3. Specify the v8-profiler package to use. Choose the relevant package from the v8-profiler package drop-down list or click the button next to it and choose the package in the dialog box that opens.
   4. Specify the port through which PhpStorm communicates with the profiler, namely, sends a command to take a snapshot when you click the Take Heap Snapshot button on the toolbar of the Run tool window.

Collecting memory profiling information

1. Select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button .
2. At any time during the application execution, click the Take Heap Snapshot button on the toolbar of the Run tool window.
3. In the dialog box that opens, choose the folder to store the taken snapshot in and specify the name to save the snapshot file with. To start analyzing the snapshot immediately, select the Open snapshot check box.
4. Click OK to save the snapshot.

Analyzing memory profiling information

The collected profiling data is displayed in the V8 Heap Tool Window which opens when you take a snapshot at choose to open it. If the window is already opened and shows the profiling data for another session, a new tab is added. Tabs that were opened automatically are named after the run configurations that control execution of the applications and collecting the profiling data.

If you want to open and analyze some previously saved mempry profiling data, choose V8 Profiling - Analyze V8 Heap Snapshot on the main menu and select the relevant `.snapshot` file. PhpStorm creates a separate tab with the name of the selected file.

The tool window has three tabs that present the collected information from difference point of views.

- The Containment tab shows the objects in you application grouped under several top-level entries: DOMWindow objects, Native browser objects, and GC Roots, which are roots the Garbage Collector actually uses. See Containment View for details.
  For each object, the tab shows its distance from the GC root, that is the shortest simple path of nodes between the object and the GC root, the shallow size of the object, and the retained size of the object. Besides the absolute values of the object's size, PhpStorm shows the percentage of memory the object occupies.
- The Biggest Objects tab shows the most memory-consuming objects sorted by their retained sizes . In this tab, you can spot memory leaks provoked by accumulating data in some global object.
- The Summary tab shows the objects in your application grouped by their types. The tab shows the number of objects of each type, their size, and the percentage of memory that they occupy. This information may be a clue to the memory state.

Each tab has a Details pane, which shows the path to the currently selected object from GC roots and the list of object’s retainers, that is, the objects that keep links to the selected object. Every heap snapshot has many “back” references and loops, so there are always many retainers for each object.

Navigating through a snapshot

- To help differentiate objects and move from one to another without losing the context, mark objects with text labels. To set a label to an object, select the object of interest and click on the toolbar or choose Mark on the context menu of the selection. Then type the label to mark the object with in the dialog box that opens.
- To navigate to the function or variable that corresponds to an object, select the object of interest and click on the toolbar or choose Edit Source on the context menu of the selection. If the button and the menu option are disabled, this means that PhpStorm has not found a function or a variable that corresponds to the selected object.
  If several functions or variables are found, they are shown in a pop-up suggestion list.
- To jump from an object in the Biggest Objects or Summary tab or Occurrences view to the same object in the Containment tab, select the object in question in the Biggest Objects or Summary tab and click on the toolbar or choose Navigate in Main Tree on the context menu of the selection. This helps you investigate the object from the containment point of view and concentrate on the links between objects.
- To search through a snapshot:
  1. In the Containment tab, click on the toolbar.
  2. In the V8 Heap Search Dialog that opens, specify the search pattern and the scope to search in. The available scopes are:
     - Everywhere: select this check box to search in all the scopes. When this check box is selected, all the other search types are disabled.
     - Link Names: select this check box to search among the object names that V8 creates when calling the C++ runtime, see http://stackoverflow.com/questions/11202824/what-is-in-javascript .
       In the V8 Heap Tool Window , link names are marked with the `%` character (`%<link name>`).
     - Class Names: select this check box to search among functions-constructors.
     - Text Strings: select this check box to perform a textual search in the contents of the objects.

- Snapshot Object IDs: select this check box to search among the unique identifiers of objects. V8 assigns such a unique identifier in the format to each object when the object is created and preserves it until the object is destroyed. This means that you can find and compare the same objects in several snapshots taken within the same session.
  In the V8 Heap Tool Window , object IDs are marked with the `@` character (`@<object id>`).
- Marks: select this check box to search among the labels you set to objects manually by clicking on the toolbar of the Containment tab.

The search results are displayed in the Details pane, in a separate Occurrences of '<search pattern>' view. To have the search results shown grouped by the search scopes you specified, press the Group by Type toggle button on the toolbar.

When you open the dialog box next time, it will show the settings from the previous search.

Using Pug (Jade) Template Engine

PhpStorm supports integration with the Pug (Jade) template engine.

On this page:

- Before you start
- Changes to the UI
- Using Pug(Jade) templates in a NodeJS application

Before you start

1. Enable Node.js support as described in Node.js .
2. Make sure the NodeJS and Pug (ex-Jade) plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Also, if you need a file watcher, make sure that the plugin File Watchers is installed and enabled.

The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Changes to the UI

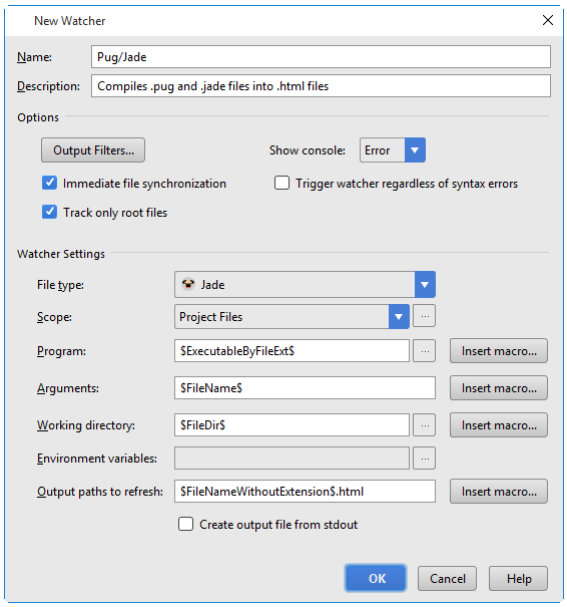
The Pug (ex-Jade) plugin introduces the following changes to the PhpStorm UI:

- The Jade file item is added to the New menu.
- The Pug files are marked with the icon ; the Jade files are marked with the icon .
- Coding assistance is provided in the Pug (Jade)-specific and HTML context:
  - Code formatting
  - Syntax highlighting
  - Code completion
  - Color schemes

Using Pug(Jade) templates in a NodeJS application

At runtime, the Pug (Jade) files fill be transformed into HTML pages.

1. Create a project from scratch, or around existing sources, or based on a NodeExpress template. See Creating New Project from Scratch , Generating a Project from a Framework Template for details.
2. Create a Pug (Jade) file. Follow these steps:
   1. In the Project tool window , select the directory in which you want to create a new file. To do that, for example, choose File | New.
   2. On the context menu, choose Jade file and specify the file name in the dialog box that opens.
3. Create a File Watcher to transform files with the extension `.jade` or `.pug` into `.html` pages:
   1. Click the Add Watcher link in the upper right-hand corner of the editor.
   2. In the New Watcher Dialog , accept the default predefined settings.
      Note that if the executable is in the PATH, then you should not specify it explicitly. Depending on the file extension (`.jade` or `.pug`), the corresponding executable is invoked.



4. As you edit a `.pug` / `.jade` file, PhpStorm invokes the file watcher which creates an `.html` file with the name of the processed `.pug` / `.jade` file and stores the generated `html` code in it.

See Using File Watchers for details.

Running NPM Scripts

On this page:

- Introduction
- Before you start
- Building a tree of scripts
- Running npm scripts from the tree of scripts
- Running tasks according to a run configuration
- Running npm scripts automatically
- Running a script as a as a before-launch task

Introduction

PhpStorm provides interface for running npm scripts . npm scripts support involves:

- Parsing `package.json` files, recognizing definitions of scripts.
- Building trees of scripts.
- Navigation between a script in the tree and its definition in the `package.json` file.
- Running and debugging scripts.
- Configuring the script execution mode and output.

Before you start

1. Download and install Node.js which contains npm .
2. Install and enable the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. Create a `package.json` file with the `scripts` property with the definitions of the scripts to run.
4. To enable debugging a script, add the `$NODE_DEBUG_OPTION` to its definition in the `package.json` file, for example:

```
{
                    "name": "application-name",
                    "version": "0.0.1",
                    "scripts": {
                    "main": "node $NODE_DEBUG_OPTION ./app-compiled.js"
                    }
                    }
```

Scripts are launched in four ways:

- From a tree of scripts in the dedicated NPM Tool Window . The tool window opens when you invoke npm by choosing Show npm Scripts on the context menu of a `package.json` in the Project tool window or of a `package.json` opened in the editor.
- According to a dedicated run configuration, see Run/Debug Configuration: NPM .
- Automatically, as a start-up task.
- As a before-launch task, from another run configuration.

The result of executing a script is displayed in the Run tool window . The tool window shows the npm script output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed script is displayed on the title bar of the tool window.

Building a tree of scripts

- If the npm tool window is not opened yet, do one of the following:
  - Select the required `package.json` file in the Project tool window and choose Show npm Scripts on the context menu of the selection.
  - Open the required `package.json` file in the editor and choose Show npm Scripts on the context menu of the editor.

  In either case, the npm tool window opens showing the scripts tree built according to the selected or opened `package.json` file.
- If the npm tool window is already opened, click on the toolbar and choose the required `package.json` file from the list. PhpStorm adds a new node and builds a scripts tree under it. The title of the node shows the path to the `package.json` file according to which the tree is built.
- To re-build a tree, switch to the required node and click on the toolbar.

Running npm scripts from the tree of scripts

- To run a script, do one of the following:
  - Double click the required script in the tree.
  - Select the required script and choose Run <script name> on the context menu of the selection.
  - Select the required script and press `Enter`.
- To run several scripts, use the multiselect mode: hold `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys and select the required scripts, then choose Run on the context menu of the selection.

Running tasks according to a run configuration

Besides using temporary run configurations that PhpStorm creates automatically, you can create and launch your own npm run configurations. For details about run configurations, see Run/Debug Configuration and Creating and Editing Run/Debug Configurations .

To create an npm run configuration:

1. Choose Run | Edit Configuration on the main menu. Alternatively, click `Shift+Alt+F10` and select Edit Configuration from the pop-up menu.
2. Click the Add New Configuration button on the toolbar and select npm from the pop-up list.

3. In the Run/Debug Configuration: NPM dialog box that opens, specify the name of the run configuration, the npm command line command to execute, the scripts to run (use blank spaces as separators), the location of the `package.json` file to retrieve the definitions of the scripts from, and the command line arguments to execute the script with.
   Specify the location of the Node executable file and the NodeJS-specific options to be passed to this executable file, see Node parameters for details.

   If applicable, specify the environment variables for the NodeJS executable file.

To run a script according to a run configuration, select the run configuration from the list on the main tool bar and then choose Run | Run <configuration name> on the main menu or click the Run toolbar button ▶. The output is displayed in the Run tool window .

Running npm scripts automatically

If you have some scripts that you run on a regular basis, you can add the corresponding run configurations to a list of startup tasks. The tasks will be executed automatically on the project start-up.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Startup Tasks under Tools.
2. On the Startup Tasks page that opens, click + on the toolbar.
3. From the drop-down list, choose the required npm run configuration. The configuration is added to the list.
   If no applicable configuration is available in the project, click + and choose Edit Configurations. Then define a configuration with the required settings in the Run/Debug Configuration: NPM page that opens. When you save the new configuration it is automatically added to the list of startup tasks.

Running a script as a as a before-launch task

1. Open the Run/Debug Configurations dialog by choosing Run | Edit Configurations on the main menu, and select the required configuration from the list or create it anew by clicking + and choosing the relevant run configuration type.
2. In the dialog box that opens, click + in the Before launch area and choose Run npm script from the drop-down list.
3. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
   Specify the location of the Node.js interpreter and the parameters to pass to it.

Installing and Removing External Software Using Node Package Manager

On this page:

- Introduction
- Installing Node.js and Node Package Manager (npm)
- Installing an external tool globally
- Installing an external tool in a project
- Installing an external tool as a development dependency

Introduction

A number of tools are started through Node.js, for example, the CoffeeScript, TypeScript, and Less transpilers, YUI, UglifyJS, and Closure compressors, Karma test runner, Grunt task runner, etc. The Node Package Manager (npm) is the easiest way to install these tools, the more so that you have to install Node.js anyway.

Depending on the desired location of the tool executable file, choose one of the following methods:

- Install the tool globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the tool in a specific project and thus restrict its use to this project.
- Install the tool in a project as a development dependency .

In either installation mode, make sure that the parent folder of the tool is added to the `PATH` variable. This enables you to launch the tool from any folder.

Installing Node.js and Node Package Manager (npm)

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The tool is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the tool.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the tool and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing an external tool globally

Global installation makes a tool available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the tool is automatically added to the `PATH` variable, which enables you to launch the tool from any folder. To install the tool globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g  <tool name>
     ```

     The `-g` key makes the tool run in the global mode. Because the installation is performed through NPM, the tool is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the tool from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the tool, see https://npmjs.org/package/ .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing an external tool in a project

Installing a tool in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

     ```
     npm install  <tool name>
     ```
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.

3. In the Available Packages dialog box that opens, select the required package.
4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the tool in a `node_modules` folder. If your project already contains such folder, the tool is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the tool in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the tool is installed there.

In either case, make sure that the parent folder of the tool is added to the `PATH` variable. This enables you to launch the tool from any folder.

Installing an external tool as a development dependency

If a tool is a documentation or a test framework, which are of no need for those who are going to re-use your application, it is helpful to have it excluded from download for the future. This is done by marking the tool as a development dependency , which actually means adding the tool in the `devDependencies` section of the `package.json` file.

With PhpStorm, you can have a tool marked as a development dependency right during installation. Do one of the following:

– Run the installation from the command line:

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install --dev  <tool name>
```

– Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the package.
4. Select the Options check box and type `--dev` in the text box next to it.
5. Optionally specify the product version and click Install Package to start installation.

After installation, a tool is added to the `devDependencies` section of the `package.json` file.

Testing RESTful Web Services

On this page:

- Introduction
- Composing and submitting a test request to a Web service method
- Viewing and analyzing responses from Web services
- Working with cookies
- Configuring Proxy settings
- Reusing requests

Introduction

With PhpStorm, you can test RESTful Web services. PhpStorm provides facilities to emulate interaction with a WebService by composing and running requests, as if you were the application that actually calls the service. In addition to this, you can create, save, edit, and remove cookies, both received through responses and created manually. The name and value of a cookie is automatically included in each request to the URL address that matches the domain and path specified for the cookie, provided that the expiry date has not been reached. .

Testing RESTful Web Services is supported via the REST Client bundled plugin, which is by default enabled. If not, activate it in the Plugins page of the Settings dialog box.

There are two main use cases when you need to compose and run requests to a RESTful Web service:

- When you have developed and deployed a RESTful Web service and want to make sure it works as expected: that it is accessible in compliance with the specification and that it responds correctly.
- When you are developing an application that addresses a RESTful Web service. In this case it is helpful to investigate the access to the service and the required input data before you start the development. During the development, you may also call the Web service from outside your application. This may help locate errors when your application results in unexpected output while no logical errors are detected in your code and you suspect that the bottleneck is the interaction with the Web service.

Testing a RESTful Web service includes the following checks:

- That URL addresses are constituted correctly based on the service deployment end-point and the method annotations.
- That the generated server requests call the corresponding methods.
- That the methods return acceptable data.

PhpStorm enables you to run these checks from the REST Client tool window by composing and submitting requests to the local server, viewing and analyzing server responses .

If necessary, configure the Proxy settings on the HTTP Proxy page of the Settings dialog box.

Composing and submitting a test request to a Web service method

To compose and submit a test request to a Web service method

1. If you are going to test your own Web service, make sure it is deployed and running.

2. Choose Tools | Test RESTful Web Service. The REST Client dedicated tool window opens.

3. To have PhpStorm generate an authentication header which will be used in basic authentication , click the Generate Authorization Header button and in the dialog box that opens, specify your user name and password for accessing the target RESTful Web service through. Based on these credentials PhpStorm will generate an authentication header which will be used in basic authentication . Learn more at http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm .

4. Select the test request method from the HTTP method drop-down list. The available options are:
   - GET
   - POST
   - PUT
   - PATCH
   - DELETE
   - HEAD
   - OPTIONS

5. Provide the data to calculate the URL address of the target method:
   1. In the Host/port text box, type the URL address of the host where the Web service is deployed.
   2. In the Path field, specify the relative path to the method to invoke.

   You can enter the entire URL address of a method to test in the Host/port text box. Regardless of the chosen HTTP method, upon pressing ▶ PhpStorm will split the URL address into the host/port and the path to the method. The extracted relative path will be shown in the Path text box and the extracted parameters will be added to the list in the Request Parameters pane of the Request tab.

6. In the Header data pane, specify the technical data included in the request header . These data are passed through header fields and define the format of the input parameters (accept field), the response format (content-type field), the caching mechanism (cache-control field), etc.
   To add a field to the list, click Add +, then specify the field name in the Name text box and the field value in the Value drop-down list.

   The set of fields and their values should comply with the Web service API. In other words, the specified input format should be exactly the one expected by the Web service as well as the expected response format should be exactly the one that the service returns.

   For `accept`, `content-type`, and some other fields PhpStorm provides a list of suggested values. Choose the relevant format type from the Value drop-down list.

7. Create a set of parameters to be passed to the target method and specify their values. Depending on the chosen request method, you can create a list of parameters in two ways:
   - For `GET` requests, specify the parameters to be passed as a query string inside the URL. Use the Request Parameters pane. By default, the pane shows

an empty list with one line.
  - To add a parameter, click Add, then specify the name of the parameter in the Name text box and the value of the parameter in the Value drop-down list.
  - To delete a parameter from the list, select it and click Remove.
  - To suppress sending the specified query string parameters and disable the controls in the Request Parameters pane, press the Don't send anything toggle button.
- To have the parameters passed to the target method inside a request message body, use the Request Body pane or have them inserted in the request from a local file. The Request Body pane is disabled when the `GET`, `DELETE`, `HEAD`, or `OPTIONS` request method is selected.
  - To specify the parameters explicitly, choose the Text option and type the parameters and values in the text box.
  - To have the parameters inserted from a text file, choose the File contents option and specify the file location in the File to send field.
  - To have a binary file converted and sent in the request, choose the File upload(multipart/form-data) option and specify the file location in the File to send field.

In either case, the set of parameters and their types should comply with the Web service API, in particular, they should be exactly the same as the input parameters of the target method.

8. To submit a request to the server, click the Submit request button.
   Note that the server may lack certificate, or be untrusted.

> **Note** If a server is not trusted, PhpStorm shows a dialog box suggesting you to accept the server, or reject it. If you accept the server as trusted, PhpStorm writes its certificate to the trust store. On the next connect to the server, this dialog box will not be shown.

Viewing and analyzing responses from Web services

To view and analyze responses from a Web service

- To view the response to the server request, switch to the Response tab. The tab is opened automatically when a response is received. By default, the server response is shown in the format, specified in the request header through the content-type field.
- To have the response converted into another format and opened in a separate tab in the editor, use the View as HTML, View as XTML, or View as JSON buttons.
- To view the technical data provided in the header of a Web service response, switch to the Response Headers tab.

Working with cookies

Using the dedicated Cookies tab, you can create, save, edit, and remove cookies, both received through responses and created manually. The name and value of a cookie is automatically included in each request to the URL address that matches the domain and path specified for the cookie, provided that the expiry date has not been reached.

The tab shows a list of all currently available cookies that you received through responses or created manually. The cookies are shown in the order they were added to the list. When you click a cookie, its details become editable and are displayed in text boxes. See REST Client Tool Window for details.

- No specific steps are required to save a cookie received through a response, all the cookies received from servers are saved automatically. To edit a received cookie, click the row with the cookie in the list and update the details that are now shown in editable text boxes.
- To create a cookie manually, click and specify the following:
  - The name and value of the cookie to be included in requests.
  - The domain and path the requests to which must be supplied with the name, value, and expiry date of the cookie.
- To remove a cookie from the list, select the row with the cookie and click.

Configuring Proxy settings

To configure Proxy settings, follow these steps

1. Click the Configure HTTP Proxy icon.
2. In the Proxy dialog that opens, specify the following:
   - Enter the Proxy host name and Proxy port number in the Proxy host and Proxy port text boxes respectively.
   - To enable authorization, check the Use authorization check box and type the User name and password in the relevant fields.

Reusing requests

During a PhpStorm session, PhpStorm keeps track or requests and you can run any previously executed request. You can also save the settings of a request in an XML file so they are available in another PhpStorm session. When necessary, you can retrieve the saved settings and run the request again.

To rerun a request within a PhpStorm session

1. Click the Replay Recent Requests button.
2. From the Recent Requests pop-up list, select the relevant request. The fields are filled in with the settings of the selected request.
3. Click the Submit Request button.

To save the settings of a request so they can be retrieved in another PhpStorm session

- Click the Export Request button. In the dialog box that opens, specify the name of the file to save the settings in and its parent folder.

To run a request saved during a previous PhpStorm session

1. Click the Import Request button.
2. In the dialog box that opens, select the relevant XML file. The fields are filled in with the settings of the selected request.
3. Click the Submit Request button.

TypeScript Support

In this section:

- TypeScript Support
  - Introduction
  - Preparing for TypeScript development
  - Coding assistance
- Transpiling TypeScript to JavaScript
- Running TypeScript
- Debugging TypeScript
- Using TSLint Code Quality Tool

Introduction

PhpStorm supports developing and running TypeScript source code. PhpStorm recognizes `*.ts` files, and allows you to edit them providing full range of coding assistance without any additional steps from your side. TypeScript files are marked with the icon.

To run, debug, and test your code you will need it translated into JavaScript which requires a transpiler.

Preparing for TypeScript development

1. Make sure the JavaScript Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
2. Make sure the Node.js plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Coding assistance

TypeScript support includes:

- Coding assistance:
  - Code completion for keywords, labels, variables, parameters, and functions.
  - Error and syntax highlighting.
  - Code formatting and folding .
  - Numerous code inspections and quick-fixes .

  This assistance can be provided either by PhpStorm itself, or based on the data from the built-in TypeScript compiler, or through integration with the TypeScript Language Service . Provision of coding assitance is configured on the TypeScript page of the Settings / Preferences Dialog :

  - Select the Use TypeScript Service check box to get native support from the TypeScript Language Service according to the up-to-date specifications. In this case syntax and error highlighting is performed based on the annotations retrieved from the TypeScript Language Service while code completion lists contain both suggestions from the TypeScript Language Service and suggestions calculated by PhpStorm itself.
  - Select the Enable TypeScript Compiler check box to active the built-in compiler. When the check box is selected, syntax highlighting and code completion are provided based only on the data from the built-in Typescript compiler. After you select this check box, the fields below on the page become active and you can configure the behaviour of the compiler.
- Refactoring:
  - Common refactoring procedures, such as extract method , inline, rename/move, etc.
  - TypeScript-specific refactoring procedures, such as change signature, extract parameter, extract variable.

  See JavaScript-Specific Refactorings for details.
- Code generation
  - Generating code stubs based on file templates during file creation.
  - Ability to create line and block comments (`Ctrl+Slash` / `Ctrl+Shift+Slash`).
  - Downloading stubs for TypeScript definition files and configuring them as a PhpStorm JavaScript library (see Configuring JavaScript Libraries).
  - Generating `import` statements for modules, classes, and any other symbol that can be exported and called as a type, see Creating Imports , section Importing TypeScript Symbols.
    You can also configure the type of quotes to be used in `import` statements by choosing Double quotes or Single quotes from the String literal type drop-down list on the Code Style. TypeScript page of the Settings dialog box.
  - Configuring automatic insertion or skipping the public access modifier in generated code.
- Navigation and search through the source code:
  - Navigating with Structure View .
  - Navigate | Declaration (`Ctrl+B`).
  - Navigate | Implementation (`Ctrl+Alt+B`) from overridden method / subclassed class.
- Transpiling to JavaScript for further running, debugging, and testing, see Running TypeScript and Debugging TypeScript .

Transpiling TypeScript to JavaScript

On this page:

- Introduction
- Configuring and activating the built-in TypeScript compiler
- Transpiling TypeScript using the built-in compiler
- Running the built-in TypeScript compiler as a before-launch task
- Transpiling TypeScript using a file watcher
  - Before you start
  - Installing the TypeScript transpiler globally
  - Installing the TypeScript transpiler in a project
  - Creating a file watcher
  - Transpiling code

Introduction

TypeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, TypeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

In addition to this, transpilation also can involve generation of source maps that set correspondence between lines in your TypeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.

With PhpStorm, you can have your TypeScript code transpiled into JavaScript in two ways: using the built-in TypeScript compiler or through integration with the typescript tool configured as a file watcher.

Configuring and activating the built-in TypeScript compiler

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Typescript under Languages & Frameworks.
2. On the TypeScript Compiler page that opens, select the Enable TypeScript Compiler check box to activate the compiler. When the check box is selected, syntax highlighting and code completion are provided based only on the data from the built-in Typescript compiler. After you select this check box, the fields below on the page become active and you can configure the behaviour of the compiler.
3. To have the compiler "wake up" upon any change to a TypeScript file, select the Track changes check box.
   When this check box is cleared, the built-in compiler ignores changes to TypeScript files. To re-activate the compiler, open the TypeScript Compiler Tool Window (View | Tool Windows | TypeScript Compiler), and click the Compile All button on the toolbar.

   If you have not opened the TypeScript Compiler Tool Window yet and it is not available from the View menu, choose Help | Find Action, then find and launch the TypeScript Compile All action from the list.

4. In the other fields of the TypeScript Compiler page, specify the following:
   - In the Node interpreter field, specify the location of the Node.js executable file. In most cases, PhpStorm detects the Node.js executable and fills in the field automatically.
   - In the TypeScript version area, PhpStorm displays the default bundled version of the built-in compiler.
     To use a custom version of the compiler:
     1. Download the `typescriptServices.js`, `lib.d.ts`, and `lib.es6.d.ts` files from https://github.com/Microsoft/TypeScript/ .
     2. Click Edit in the Compiler version area.
     3. In the Configure TypeScript Compiler dialog box that opens, choose Custom directory and specify the folder where the downloaded files are stored.
   - From the Scope drop-down list, choose the scope to apply the compiler in. The available options are:
     - Project Files: all the files within the project content roots (see Content Root and Configuring Content Roots).
     - Project Production Files: all the files within the project content roots excluding test sources.
     - Project Test Files: all the files within the project test source roots.
     - Open Files: all the files that are currently opened in the editor.

     VCS Scopes: these scopes are only available if your project is under version control .
     - Changed Files: all changed files, that is, all files associated with all existing changelists .
     - Default: all the files associated with the changelist `Default`.

     Alternatively, click the Browse button and configure a custom scope in the Scopes dialog box that opens. For more details on scopes, see the pages Scopes and Scopes dialog .
   - Choose the Set options manually to configure transpilation manually:
     - In Command line options field, specify the command line options to be passed to the compiler. See the list of acceptable options at TSC arguments . Note that, the `-w` or `--watch` option (Watch input files) is irrelevant.
     - Select the Generate source maps check box to generate source maps that set correspondence between lines in your TypeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.
     - Select the Compile main file only check box to have PhpStorm compile only a specific file and the files that are referenced from it and ignore all the other files in the project. This may be helpful if you have a dedicated `main.ts` file which only references other files.
     - Select the Use output path check box to have the built-in compiler store the generated JavaScript files and source map in a custom folder. Specify the path to this folder explicitly or use one of the listed available macros in the format: `$<macros_name>$` . The available macros are:
       - `FileDir` : the path to the folder where the file is stored.
       - `FileRelativeDir` : the path to the file directory relative to the project root.
       - `FileDirRelativeToProjectRoot` : the path to the file directory relative to the project root.
       - `ModuleFileDir` : the path to the project root folder.
       - `SourcePath` : the path to the source folder under the content root to which the file belongs, see Configuring Content Roots and Configuring Folders Within a Content Root .

- `FileDirRelativeToSourcePath` : the path to the file relative to the source folder under the content root to the file belongs.

- Choose the Use tsconfig.json option, have the built-in compiler analyze the code according to the settings specified in the `tsconfig.json` file .

Transpiling TypeScript using the built-in compiler

1. To activate a previously configured built-in transpiler, select the Enable TypeScript Compiler check box on the TypeScript Compiler page .
   Alternatively, use the buttons on the pane which PhpStorm displays when you open a TypeScript file.
   - To enable an already configured compiler, click Enable.
   - To configure a compiler, click Configure and specify the settings on the TypeScript Compiler page that opens.
   - To leave the compiler disabled and suppress showing the pane in the future, click Dismiss, whereupon the compiler can be later activated only on the TypeScript Compiler page.

2. View the errors detected during compilation, compile separate files and the entire project in the TypeScript Compiler Tool Window . If you have configured the compiler to track changes, which is the default behaviour, the tool window opens automatically as soon as you start editing a TypeScript file.
   If tracking changes is disabled and you have closed the tool window, to re-open it choose View | Tool WIndows | TypeScript Compiler.
   - Click the Compile Current File toolbar button to run the compiler against the entire file opened in the active editor tab. This approach is helpful if you do not want the compiler to wake up on any change as you edit your code and for this purpose the Track Changes check box on the TypeScript page of the Settings dialog box is cleared.
     If you click this button from the Project Errors pane, after compilation PhpStorm switches to the Current Errors pane.
   - Click the Compile All toolbar button to run the compiler against all the TypeScript files in the current project. If you click this button from the Current Errors pane, after compilation PhpStorm switches to the Project Errors pane.
   - To clear the contents of the current pane, click the Clear All toolbar button .
   - To navigate to the fragment of code where an error occurred during compilation, select the error message in question and choose Jump to Source on the context menu of the selection.

The compiler stores the generated output in a separate file. The JavaScript file has the name of the source TypeScript file and the extension `js` , the source map file has the name of the source TypeScript file and the `js.map` extension. In the Project Tree, the files are shown under the source TypeScript file which is now displayed as a node. This presentation is preserved even if you turn off the compiler during a PhpStorm session.

Running the built-in TypeScript compiler as a before-launch task

You can run the built-in compiler before launching a run configuration to make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files.

1. Check whether the compiler is activated on the TypeScript Compiler page , see Transpiling TypeScript to JavaScript above.
2. Open the Run/Debug Configurations dialog by choosing Run | Edit Configurations on the main menu, and select the required configuration from the list or create it anew by clicking + and choosing the relevant run configuration type.
3. In the dialog box that opens, click + in the Before launch area and choose Compile TypeScript from the drop-down list.
4. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
   - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
   - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.

Transpiling TypeScript using a file watcher

Using file watchers is recommended if your project consists of several independent parts and you want to transpile them separately. In this case you can configure several file watchers each with its own scope and apply these file watchers independently.

PhpStorm supports integration with the typescript tool. The TypeScript transpiler translates TypeScript code into JavaScript and creates source maps that set correspondence between lines in your TypeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.

In PhpStorm, transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files, you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

The easiest way to install the TypeScript transpiler is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the TypeScript transpiler executable file, choose one of the following methods:

- Install the transpiler globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the transpiler in a specific project and thus restrict its use to this project.
- Install the transpiler in a project as a development dependency .

In either installation mode, make sure that the parent folder of the TypeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

Before you start

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The TypeScript transpiler is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the TypeScript transpiler.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the TypeScript transpiler and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

3. Make sure the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If it is not, enable the plugin. See Enabling and Disabling Plugins for details.

Installing the TypeScript transpiler globally

Global installation makes a transpiler available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the transpiler is automatically added to the `PATH` variable, which enables you to launch the transpiler from any folder. To install the transpiler globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g typescript
     ```

     The `-g` key makes the transpiler run in the global mode. Because the installation is performed through NPM, the TypeScript transpiler is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the TypeScript transpiler, see https://npmjs.org/package/typescript .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing the TypeScript transpiler in a project

Installing the transpiler in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

     ```
     npm install typescript
     ```
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package.
  4. Optionally specify the product version and click Install Package to start installation.

Project level installation is helpful and reliable in template-based projects of the type Node Boilerplate or Node.js Express, which already have the `node_modules` folder. The latter is important because NPM installs the TypeScript transpiler in a `node_modules` folder. If your project already contains such folder, the TypeScript transpiler is installed there.

Projects of other types or empty projects may not have a `node_modules` folder. In this case npm goes upwards in the folder tree and installs the TypeScript transpiler in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be outside your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the TypeScript transpiler is installed there.

In either case, make sure that the parent folder of the TypeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

Creating a file watcher

PhpStorm provides a common procedure and user interface for creating File Watchers of all types. The only difference is in the predefined templates you choose in each case.

1. To start creating a File Watcher, open the Settings/Preferences dialog box by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X on the main menu, and then click File Watchers under the Tools node. The File Watchers page that opens, shows the list of File Watchers that are already configured in the project.
2. Click the Add button + or press `Alt+Insert` and choose the TypeScript predefined template from the pop-up list.
3. In the Program text box, specify the path to the executable file `tsc.cmd` or `tsc` in the `node_modules\.bin` folder.
   - If you installed TypeScript in the current project, the file is stored in the `node_modules\.bin` folder under the project root.
   - In case of global installation, the executable file is stored in the `node_modules\.bin` folder in the user home.

   Type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.

4. Proceed as described on page Using File Watchers .

Transpiling code

When you open a TypeScript file, PhpStorm checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, PhpStorm displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, PhpStorm starts it automatically upon the event specified in the New Watcher dialog .

- If the Immediate file synchronization check box is selected, the File Watcher is invoked as soon as any changes are made to the source code.
- If the Immediate file synchronization check box is cleared, the File Watcher is started upon save (File | Save All, `Ctrl+S`) or when you move focus from PhpStorm (upon frame deactivation).

The transpiler stores the generated output in a separate file. The file has the name of the source TypeScript file and the extension `js` or `js.map` depending on the transpiler type. The location of the generated files is defined in the Output paths to refresh text box of the New Watcher dialog . Based on this setting, PhpStorm detects the transpiler output. However, in the Project Tree, they are shown under the source `.ts` file which is now displayed as a node. This presentation is preserved even if you turn off the file watcher during a PhpStorm session.

Running TypeScript

On this page:

- Introduction
- Running a file with injected TypeScript from PhpStorm

Introduction

TypeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, TypeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

For more details about using transpilers in PhpStorm, see the section Using File Watchers .

Running a file with injected TypeScript from PhpStorm

Note that no run configuration is required for launching applications with injected TypeScript from PhpStorm.

1. Transpile the TypeScript code into Javascript .
2. In the editor, open the HTML file with the TypeScript reference. This HTML file does not necessarily have to be the one that implements the starting page of the application.
3. Manually replace the reference to the TypeScript file with the reference to the JavaScript file that was generated through the transpilation.
4. Do one of the following:
   - Choose View | Preview file in on the main menu or press `Alt+F2`. Then select the desired browser from the pop-up menu.
   - Hover your mouse pointer over the code to show the browser icons bar: Click the icon that indicates the desired browser.

Debugging TypeScript

TypeScript code is not processed by browsers that work with JavaScript code. Therefore to be executed, TypeScript code has to be translated into JavaScript. This operation is referred to as transpilation and the tools that perform it are called transpilers.

Because JavaScript debugging is supported only in Google Chrome and other browsers of the Chrome family, TypeScript can also be debugged only in these browsers.

For more details about using transpilers in PhpStorm, see the section Using File Watchers .

To debug TypeScript code:

1. Transpile the TypeScript code into Javascript using the built-in compiler or a TypeScript file watcher, see Transpiling TypeScript to JavaScript for details.
2. In the editor, open the HTML files with the TypeScript references.
3. Manually replace the references to the TypeScript files with the references to the JavaScript files that were generated through the transpilation.
4. Configure the JavaScript debugger.
5. Configure and set breakpoints in the TypeScript code. Although actually debugging will be performed against the generated JavaScript, PhpStorm will emulate TypeScript debugging using the generated source-maps that set correspondence between lines in the TypeScript and in the JavaScript code.
6. Proceed as if you were debugging JavaScript:
   1. Initiate a debugging session. For details about local and remote debugging, about debugging single files and entire applications, see Debugging JavaScript .
   2. If you are using Chrome, install and configure the JetBrains Chrome Extension. For details, see Using JetBrains Chrome Extension .
   3. Step through the program , stop and resume program execution, examine it when suspended , etc.

Using TSLint Code Quality Tool

On this page:

- Introduction
- Before you start
- Installing TSLint
- Activating and configuring the TSLint tool

Introduction

PhpStorm provides facilities to run TypeScript-specific code quality inspections through integration with the TSLint code verification tool. This tool registers itself as a PhpStorm code inspection: it checks TypeScript code for most common mistakes and discrepancies without running the application. When the tool is activated, it launches automatically on the edited TypeScript file. Discrepancies are highlighted and reported in pop-up information windows, a pop-up window appears when you hover the mouse pointer over a stripe in the Validation sidebar. You can also press `Alt+Enter` to examine errors and apply suggested quick fixes. Learn more about inspections and intention actions at Code Inspection and Intention Actions .

Before you start

1. The TSLint tool is run through NodeJS, therefore make sure the NodeJS runtime environment is downloaded and installed on your computer. The runtime environment also contains the Node Package Manager(npm) through which TSLint is installed.
2. Integration with NodeJS and NPM is supported through the NodeJS plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing TSLint

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
3. In the Available Packages dialog box that opens, select the `tslint` package and click Install Package.

Learn more about installing tools through NPM in Installing and Removing External Software Using Node Package Manager .

Activating and configuring the TSLint tool

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click TSLint under TypeScript.
2. On the TSLint page that opens, select the Enable check box. After that all the controls in the page become available.
3. Specify the location of the Node.js executable file and the path to the tslint package.
4. In the Configuration File area, appoint the configuration to use. By default, PhpStorm first looks for a `tslint.json` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. If no `tslint.json` file is found, TSLint uses its default embedded configuration file. Accordingly, you have to define the configuration to apply either in a `tslint.json` configuration file, or in a custom JSON configuration file, or rely on the default embedded configuration.
   - To have PhpStorm look for a `tslint.json` file, choose the Search for tslint.json option. If no `tslint.json` file is found, the default embedded configuration file will be used.
   - To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the ... button and select the relevant file from the dialog box that opens.
5. If necessary, in the Additional Rules Directory field, specify the location of the files with additional code verification rules. These rules will be applied after the rules from `tslint.json` or the above specified custom configuration file and accordingly will override them.

TextMate

In this section:

- TextMate
  - Prerequisites
  - TextMate support
- Importing TextMate Bundles
- Editing Files Using TextMate Bundles
- Tutorial: Using TextMate Bundles

Prerequisites

Before you start working with TextMate, make sure that the TextMate bundle support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Also, make sure that the TextMate bundles are downloaded to your computer.

TextMate support

TextMate files are marked with .

TextMate support in PhpStorm includes:

- Possibility to import bundles .
- Possibility to establish mappings between the color schemes of PhpStorm and TextMate.
- Syntax and error highlighting .
- Code completion in `*.markdown` files.

Importing TextMate Bundles

TextMate bundles become available to PhpStorm when they are downloaded to your computer.

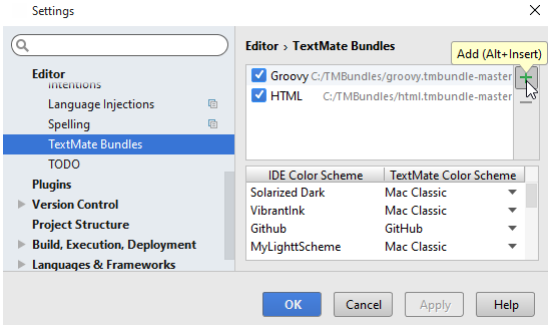
To import a TextMate bundle, follow these general steps

1. Make sure that TextMate bundles you want to import to PhpStorm, are already downloaded. For example, you can find the desired TextMate bundles on GitHub .

   Note It is important to note that PhpStorm cannot read the binary bundles. Use only bundle sources instead. You can find them on GitHub .

2. Open Settings/Preferences dialog , and click TextMate Bundles.

3. In the TextMate Bundles page, click +:



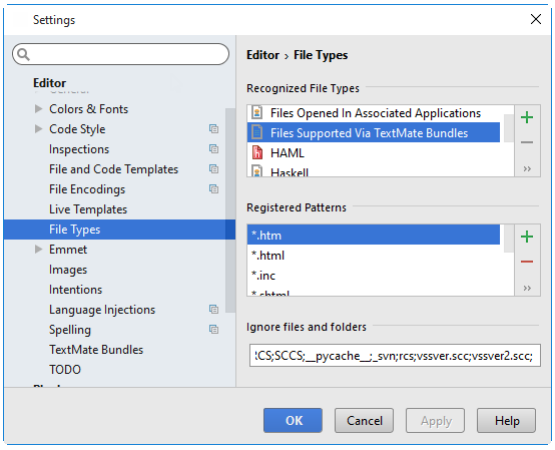
4. In the Select Path dialog box that opens, locate the desired bundle in the file system, and click OK. Repeat this step as required.

5. Apply changes.

6. If necessary, add the desired file type to the to the imported TM bundle.
   For example, if you want Ruby files with `.rb` extension to be supported by the PhpStorm's TextMate integration, you have to open for editing the file `Ruby.tmbundle\Syntaxes\Ruby.plist`, locate the section `fileTypes`, and under `array` add `rb`.

   Then restart PhpStorm.

7. Just to make sure that the desired file type will be opened via the TextMate bundles, open File Types page of the Settings/Preferences dialog, among the recognized file types find Files supported via TextMate bundles, and see the list of extensions:



Editing Files Using TextMate Bundles

1. Create a file with the extension `*.markdown` .

   Note A file with the extension `*.markdown` should be associated with the type Files supported via TextMate Bundles, as described in the section Creating and Registering File Types .

2. Open file for editing, and enter the desired text. The lines are highlighted according to the imported bundles.

Tutorial: Using TextMate Bundles

On this page:

- What this tutorial is about
- Prerequisites
- Importing bundles
- Extension conflicts
- Testing

What this tutorial is about

Projects can contain file types unknown to PhpStorm. While PhpStorm comes with the built-in support for many programming and scripting languages, you might want to have syntax highlighting for the project-specific languages. For example, a project can contain a shell script, or Perl; a configuration file can exist in a project for the infrastructure automation purposes. If you want to have syntax highlighting for these cases, use the powerful PhpStorm's integration with the text editor TextMate .

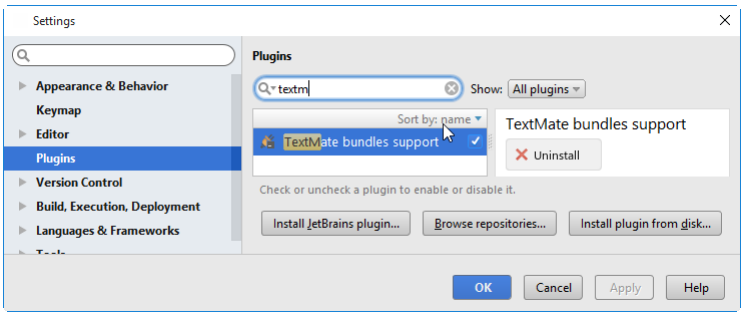
This tutorial aims to walk you step by step through configuring PhpStorm to use the TextMate Bundles, and editing files with the registered extensions.

Learning TextMate is out of scope of this tutorial. For information about TextMate, refer to the product documentation .

Prerequisites

Make sure that:

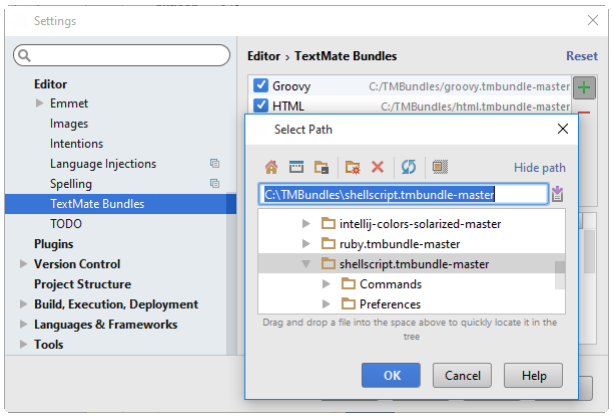
- You have already downloaded bundles you want to use. You can, for example, find the bundles you want to install on GitHub or Subversion .
- You are working with PhpStorm 2016.1 (where TextMate Bundles has been supported) or higher. In this tutorial, PhpStorm 2016.2 is used.
- Before you start working with TextMate Bundles, make sure that the TextMate bundle support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .



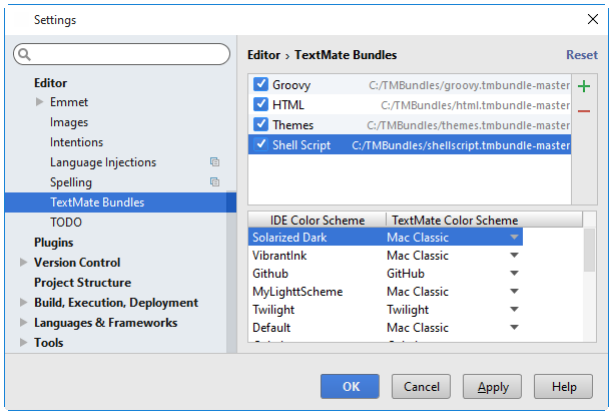
Importing bundles

Suppose you want PhpStorm to highlight syntax of the Shell Script files. For this purpose, you have already downloaded the Shell Script TextMate Bundle . It now resides on your hard disk, and you only have to import this bundle into PhpStorm.

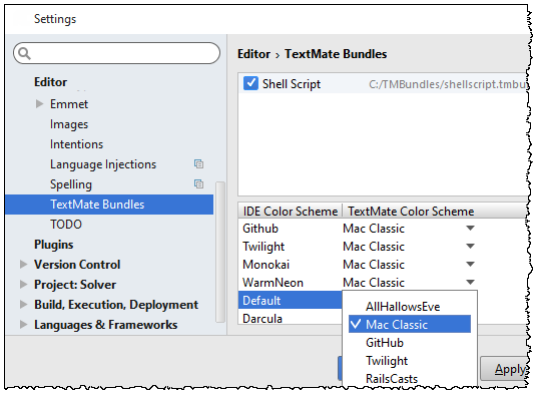
OK, off we go. On the main toolbar, click , and under the Editor node, click TextMate Bundles . Then, in the TextMate Bundles area, click +, and locate the desired bundle on your hard disk:



Click OK to apply changes. The Shell Script bundle appears in the list of recognized bundles, and its local path is visible to the right:



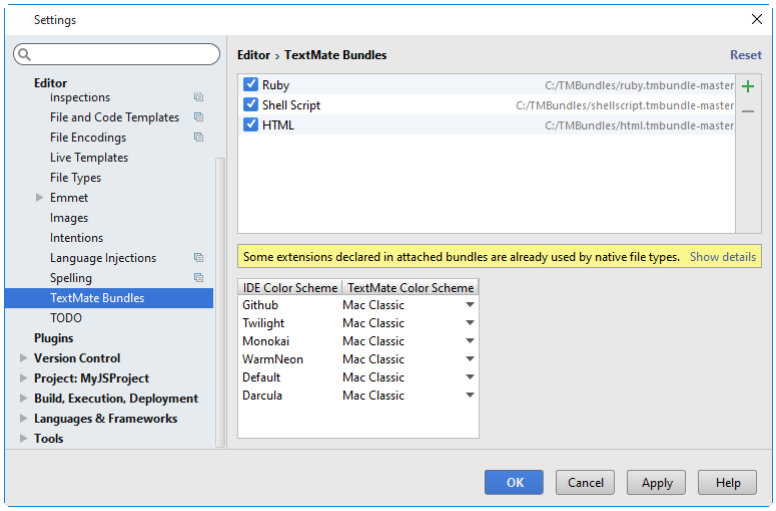
However, we have not yet defined which color scheme PhpStorm will use to highlight Shell Script syntax. As you already know, PhpStorm provides a number of color schemes, from the classic-looking ones to the fashionable dark schemes, like Darcula. Have a look at the color scheme mapping section in the lower part of the TextMate Bundles page. By default, the PhpStorm's Default color scheme maps to the Mac Classic. If we want to use a different color scheme for our bundle, we can click the "Mac Classic" cell in the table of mappings, and select the desired scheme from the list of available ones. However, let's stick to the suggested scheme:



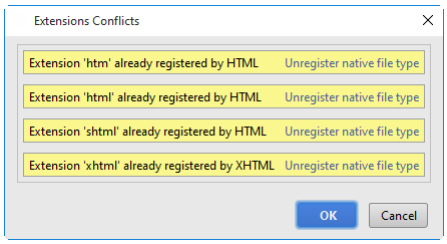
Tip If you want to use a custom TextMate color scheme, you can import a TextMate bundle with schemes, and it will become visible in the list of TextMate color schemes after clicking Apply.

Extension conflicts

Suppose you've imported a bundle that runs into a conflict with the existing file types. The conflict is immediately reported:



Clicking the Show details link opens the dialog box that gives you the chance to unregister the required extensions from the native file types:



The node Files supported via TextMate Bundles now shows the new extensions (`.htm, .html`), and the node HTML lacks these extensions:

Testing

Once a TextMate bundle is added, PhpStorm provides syntax highlighting for the file types registered with the bundle. Here's a sample script that uses the Shell Script TextMate bundle we've cloned earlier:

run.bash

```bash
#!/bin/sh
echo "This script checks the existence of..."
echo "Checking..."
if [ -f var/log/messages ]
    then
        echo "var/log/messages exist"
echo
echo "done"
fi
```

Using Ext JS Framework

PhpStorm provides you with the ability to create rich JavaScript interfaces in a browser using the Ext JS client-side framework.

To enable the use of Ext JS in your project, perform these general steps:

1. Download the Ext JS.
2. Configure it as a custom PhpStorm JavaScript library .
3. Associate the library with the project .

Using the Flow Type Checker

PhpStorm provides basic support of Flow static type checker that brings type annotations to JavaScript. This support involves recognition and syntax highlighting of Flow structures on all operating systems. If you are using Unix or OS X, you can also run Flow in the command-line mode.

For information on installing Flow, see Getting Started with Flow .

To have PhpStorm recognize Flow structures, provide correct syntax highlighting, and avoid false-positive error highlighting, change the JavaScript language level to Flow:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click JavaScript under Languages&Frameworks.
2. On the JavaScript page that opens, choose Flow from the JavaScript Language Version drop-down list.

Using Bower Package Manager

On this page:

- Introduction
- Preparing to install Bower
- Installing Bower globally
- Installing Bower in a project
- Creating a Bower configuration file bower.json
- Configuring Bower in PhpStorm

Introduction

PhpStorm provides interface for installing, uninstalling, and upgrading client-side libraries and frameworks for your project using the Bower Package Manager . Alternatively, you can use the tool in the command line mode from the embedded local terminal .

The easiest way to install the Bower package manager is to use the Node Package Manager (npm), which is a part of Node.js . See Installing and Removing External Software Using Node Package Manager for details.

Depending on the desired location of the Bower package manager executable file, choose one of the following methods:

- Install the package manager globally at the PhpStorm level so it can be used in any PhpStorm project.
- Install the package manager in a specific project and thus restrict its use to this project.
- Install the package manager in a project as a development dependency .

In either installation mode, make sure that the parent folder of the Bower package manager is added to the `PATH` variable. This enables you to launch the package manager from any folder.

PhpStorm provides user interface both for global and project installation as well as supports installation through the command line.

Preparing to install Bower

1. Download and install Node.js . The runtime environment is required for two reasons:
   - The Bower package manager is started through Node.js.
   - NPM, which is a part of the runtime environment, is also the easiest way to download the Bower package manager.

   If you are going to use the command line mode, make sure the path to the parent folder of the Node.js executable file and the path to the `npm` folder are added to the `PATH` variable. This enables you to launch the Bower package manager and npm from any folder.
2. Install and enable the NodeJS repository plugin as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Installing Bower globally

Global installation makes a package manager available at the PhpStorm level so it can be used in any PhpStorm project. Moreover, during installation the parent folder of the package manager is automatically added to the `PATH` variable, which enables you to launch the package manager from any folder. To install the package manager globally, do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where NPM is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs .
  3. Type the following command at the command line prompt:

     ```
     npm install -g bower
     ```

     The `-g` key makes the package manager run in the global mode. Because the installation is performed through NPM, the Bower package manager is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the package manager from any folder.

     For more details on the NPM operation modes, see npm documentation . For more information about installing the Bower package manager, see https://npmjs.org/package/bower .
- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages&Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package to install.
  4. Select the Options check box and type `-g` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

Installing Bower in a project

Installing a package manager in a specific project restricts its use to this project. To run project installation, do one of the following:

- Run the installation from the command line:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the project root folder and type the following command at the command line prompt:

```
npm install bower
```

- Run NPM from PhpStorm using the Node.js and NPM page of the Settings dialog box.
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Node.js and NPM under Languages & Frameworks.
  2. On the Node.js and NPM page that opens, the Packages area shows all the Node.js-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package.
  4. Optionally specify the product version and click Install Package to start installation.

Creating a Bower configuration file bower.json

1. In the command line mode, switch to your project directory.
2. Type the following command at the command line prompt:

```
bower init
```

   If Bower does not start, check the installation: the parent folder or the Bower executable file should be specified in the PATH variable.
3. Answer the questions to specify the following basic settings:
   - The testing framework to use.
   - The browsers to be captured automatically.
   - The patterns that define the location of test files to be involved in testing or excluded from it. For more details, see

   For more details, see Bower: Configuration .

Configuring Bower in PhpStorm

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Bower under JavaScript.
2. On the Bower page that opens, specify the location of the Node.js and Bower executable files and the `bower.json` configuration file.

Installing and Removing Bower Packages

Bower packages can be installed and used only in a specific project. PhpStorm supports two installation modes: from the command line and through the dedicated user interface.

On this page:

- Installing a Bower package in the command-line mode
- Installing a Bower package through the PhpStorm interface
- Removing Bower packages
- Installing a Bower package as a development dependency

Installing a Bower package in the command-line mode

1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
2. Switch to the directory where Bower is stored or define a `PATH` variable for it so it is available from any folder, see Installing Bower .
3. Type the following command at the command line prompt:

```
bower install <tool name>
```

Installing a Bower package through the PhpStorm interface

1. Run Bower from PhpStorm using the Bower page of the Settings dialog box.
   1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Bower under JavaScript.
   2. On the Bower page that opens, the Packages area shows all the Bower-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
   3. In the Available Packages dialog box that opens, select the required package.
   4. Optionally specify the product version and click Install Package to start installation.

Removing Bower packages

Open the Bower page, select the package to remove, and click –.

Installing a Bower package as a development dependency

If a Bower package is a documentation or a test framework, which are of no need for those who are going to re-use your application, it is helpful to have it excluded from download for the future. This is done by marking the tool as a development dependency , which actually means adding the Bower package in the `devDependencies` section of the `package.json` file.

With PhpStorm, you can have a Bower package marked as a development dependency right during installation. Do one of the following:

- Run the installation from the command line in the global mode:
  1. Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
  2. Switch to the directory where Bower is stored or define a `PATH` variable for it so it is available from any folder, see Installing Bower .
  3. Type the following command at the command line prompt:

```
bower install -dev <tool name>
```

- Install the package using the PhpStorm user interface:
  1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Bower under JavaScript.
  2. On the Bower page that opens, the Packages area shows all the Bower-dependent packages that are currently installed on your computer, both at the global and at the project level. Click +.
  3. In the Available Packages dialog box that opens, select the required package.
  4. Select the Options check box and type `--dev` in the text box next to it.
  5. Optionally specify the product version and click Install Package to start installation.

After installation, a Bower package is added to the `devDependencies` section of the `package.json` file.

Using ReactJS in JavaScript and TypeScript

ReactJS introduces JSX and TSX, an XML-like syntax that you can use inside your JavaScript and TypeScript code respectively.

On this page:

- Installing ReactJS
- Enabling ReactJS support in JavaScript code
- Enhancing code completion with typed parameter information
- Using ReactJS with TypeScript

Installing ReactJS

Download the ReactJS Starter Kit and extract the files from the archive.

Enabling ReactJS support in JavaScript code

PhpStorm recognizes JSX code and provides syntax highlighting, code completion, navigation, and code analysis for it. Code completion and navigation is also provided for ReactJS methods. See Auto-Completing Code and Navigating to Declaration or Type Declaration of a Symbol .

PhpStorm can also provide code completion for HTML tags and component names that you have defined inside methods in JavaScript or inside other components.

Completion also works for imported components with ES6 style syntax. You can navigate from a component name to its definition with `Ctrl+B` or see a definition in a popup with `Ctrl+Shift+I`.

In JSX tags, PhpStorm provides coding assistance for ReactJS-specific attributes such as `className` or `classID` . Moreover, for class names you can autocomplete classes defined in the project's CSS files.

Code completion is also provided for JavaScript expressions inside curly braces. Code completion applies to all methods and functions that you have defined.

PhpStorm supports Emmet in JSX code thus enabling some ReactJS-specific twists. For example, the abbreviation `div.my-class` would expand in JSX to `<div className="my-class"></div>` and not to `<div class="my-class"></div>` as it would in HTML.

To get coding assistance with JSX or JavaScript:

1. Make sure you have the `react.js` file anywhere under the project root folder, possibly copy it from the downloaded ReactJS Starter Kit after extraction. Alternatively, configure `react.js` as an external JavaScript library, see Configuring JavaScript Libraries .
2. Do one of the following:
   - Create a `.jsx` file to write your code in. PhpStorm recognizes this file type and provides coding assistance in it.
   - To use the same coding assistance in `.js` files, change the language level to JSX Harmony: Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click JavaScript under Languages&Frameworks. On the JavaScript page that opens, choose JSX Harmony from the JavaScript Language Version drop-down list.

Enhancing code completion with typed parameter information

To see information on parameters that you can use in a method, configure a TypeScript definition file react.d.ts as a JavaScript library and add it to your project:

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Languages&Frameworks node, and then click Libraries under JavaScript.
2. On the JavaScript. Libraries page that opens, click Download on the toolbar.
3. In the Download Library dialog box that opens, select react from the list of stubs, and click Download and Install. You return to the JavaScript. Libraries page where the library is added to the list as Global and is already activated.
4. Copy the `react.d.ts` file to your project.

Using ReactJS with TypeScript

PhpStorm recognizes and supports ReactJS syntax in TypeScript context (TSX) only in `.tsx` files. To use ReactJS with TypeScript code:

1. Add a TypeScript definition file react.d.ts to your project as described above in the section Enhancing Code Completion with Typed Parameter Information or add the `react.js` to your project.
2. Write your code in `.tsx` files.

Vagrant: Working with Reproducible Development Environments

In this section:

- Vagrant: Working with Reproducible Development Environments
  - Prerequisites
  - Basics
  - Vagrant support
  - Preparing to work with Vagrant
  - Initializing the Vagrantfile
  - Creating and launching an instance (virtual machine)
  - Stopping, suspending, resuming, reloading, and destroying an instance (virtual machine)
- Creating and Removing Vagrant Boxes
- Initializing Vagrant Boxes

Prerequisites

Before you start working with Vagrant, make sure that the Vagrant plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Besides that, make sure that the following prerequisites are met (outside of PhpStorm):

- Oracle's VirtualBox is installed on your computer.
- Vagrant is installed on your computer, and all the necessary infrastructure is created.
- The parent folders of the following executable files are added to the system PATH variable:
  - `vagrant.bat` or `vagrant` from your Vagrant installation. This should be done automatically by the installer.
  - `VBoxManage.exe` or `VBoxManage` from your Oracle's VirtualBox installation.
- The required virtual boxes are created.

Basics

Integration with Vagrant helps you create reproducible development environments defined by VagrantFile configuration files. To have the same environment set up on the machines of all the team members you only need to put the VagrantFile under your team version control. Any team member can set up the required environment by running the `vagrant up` command with the relevant VagrantFile. Vagrant support in PhpStorm is provided through the Vagrant plugin that comes bundled with PhpStorm .

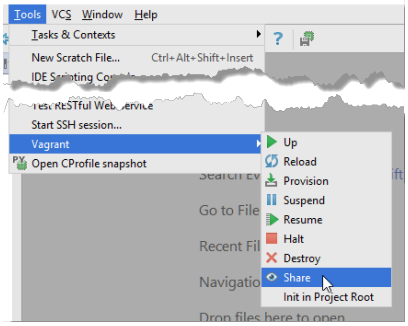
In the context of PhpStorm, the following terms are used to denote Vagrant-specific notions:

- A Vagrant box in the PhpStorm user interface or Vagrant base box in the current documentation is a box in the native Vagrant terminology. It denotes a pure image, a skeleton, on the base of which a specific environment is customized, provisioned, and deployed on your machine.
- An instance is a virtual machine. In other words, it is a specific environment customized, provisioned, and deployed on your machine on the base of a Vagrant base box and in accordance with a VagrantFile. In other similar products and documentation, an instance can be referred to as virtual machine.
- An instance folder is the folder where the relevant VagrantFile is stored after initialization and where PhpStorm will look for it. By default, it is the project root folder.

Vagrant support

Once the Vagrant plugin is enabled, the following changes are made to the PhpStorm UI:

- A Vagrant page is added to the Settings dialog.
- A Vagrant node is added to the Tools menu. The node contains the following commands that correspond to the standard Vagrant actions:



Vagrant support in PhpStorm features:

- capability to create new virtual boxes, and delete the unnecessary ones.
- capability to initialize Vagrant boxes , and execute other Vagrant commands without leaving the IDE.
- Multiple Vagrant configuration: when performing the vagrant up command, you are suggested to select the desired virtual box.

Preparing to work with Vagrant

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X, and click Vagrant under Tools. The Vagrant page opens.
2. Specify the Vagrant executable file. Because the parent folder of the executable file has been already added to the system PATH variable, type just the name of the executable.
3. In the Instance Folder field, specify the fully qualified path to the directory where the `VagrantFile` is initialized and stored. A `VagrantFile` is a configuration file that defines the instance (virtual machine) you need. The file contains the virtual IP address, port mappings, and the memory size to assign. The file can specify which folders are shared and which third-party software should be installed. According to the `VagrantFile` your instance (virtual machine) is configured, provisioned against the relevant Vagrant base box, and deployed on your computer. A `VagrantFile` is created through the `vagrant init` command.
   When creation of an instance (virtual machine) is invoked either through the `vagrant up` command or through the Tools | Vagrant | Up menu option, PhpStorm looks for

the `VagrantFile` in the directory specified in the Instance folder field. For more information, see http://docs.vagrantup.com/v2/vagrantfile/ .

You can create a `VagrantFile` in any directory and appoint it as instance folder. If the field is empty, PhpStorm will treat the project root as the instance folder and look for a `VagrantFile` in it.

4. In the Vagrant Boxes area, configure a list of the predefined Vagrant base boxes available in PhpStorm. Each item presents a Vagrant base box on which Vagrant configures and launches its instances (virtual machines). The entries of this list correspond to the output of the command `vagrant box list` .
   - To download a new base box, click the Add button +. In the dialog box that opens, specify the URL address to access the base box and the name to refer to it in PhpStorm. By default, PhpStorm suggests the URL to the lucid32 box.
     This command corresponds to `vagrant box add <name> <URL>` . As a result, the specified base box is downloaded to your machine.
   - To remove a base box, select it in the list and click the Remove button —. The base box and the nested files are physically deleted from the disk. This command corresponds to `vagrant box remove <name>` .

Initializing the Vagrantfile

A `VagrantFile` is a configuration file that defines the instance (virtual machine) you need. The file contains the virtual IP address, port mappings, and the memory size to assign. The file can specify which folders are shared and which third-party software should be installed. According to the `VagrantFile` your instance (virtual machine) is configured, provisioned against the relevant Vagrant base box, and deployed on your computer. A `VagrantFile` is created through the `vagrant init` command.

You can initialize the Vagrantfile in any folder, just keep in mind that this folder should be specified as the instance folder on the Vagrant page of the Settings dialog box. Otherwise PhpStorm will be unable to find the relevant VagrantFile during the instance (virtual machine) creation.

To initialize the VagrantFile, do one of the following:

To initialize the Vagrantfile

1. To have the Vagrantfile created in the project root, choose Tools | Vagrant | Init on Project Root on the main menu and select the target project root from the pop-up list that opens. The output of the `init` command is displayed in the Run tool window.

2. To have the Vagrantfile created in a specific folder, launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details). Then type the following commands at the prompt:

```
cd <directory to initialize the VagrantFile in>
vagrant init <base box name> <base box url>
```

Once the VagrantFile initialization is successfully completed, you are ready to import the base box, provision and deploy it according to the VagrantFile, thus creating your own instance (virtual machine).

Creating and launching an instance (virtual machine)

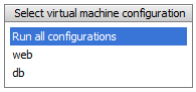
Creating an instance (virtual machine) means importing and provisioning a base box according to the VagrantFile located in the instance folder.

To create an instance, follow these steps:

1. Do one of the following:
   - To have the instance created in the project root, choose Tools | Vagrant | Up.
   - Launch the embedded Terminal by hovering your mouse pointer over in the lower left corner of PhpStorm and choosing Terminal from the menu (see Working with Embedded Local Terminal for details).
     Then type the following commands at the prompt:

```
cd <instance folder>
vagrant up
```

2. Select the desired virtual machine configuration from the suggestion list:



Stopping, suspending, resuming, reloading, and destroying an instance (virtual machine)

To reload an instance

- If you have made some changes to the VagrantFile and want a running virtual machine updated in accordance to them, choose Tools | Vagrant | Reload on the main menu or run the following command in the embedded Terminal:

```
vagrant reload
```

For details, see http://docs.vagrantup.com/v2/cli/reload.html .

To suspend an instance

- To temporary stop the operating system on a running virtual machine (guest machine) and save the exact state of the environment as it is at this moment so it can be resumed exactly from this point, choose Tools | Vagrant | Suspend on the main menu or run the following command in the embedded Terminal:

```
vagrant suspend
```

For details, see http://docs.vagrantup.com/v2/cli/suspend.html .

To resume an instance

- To resume a previously suspended operating system on a virtual machine (guest machine) and have it run from the state saved at the suspension moment, choose Tools | Vagrant | Resume on the main menu or run the following command in the embedded Terminal:

```
vagrant resume
```

For details, see http://docs.vagrantup.com/v2/cli/resume.html .

To shut an instance down

- To shut down the operating system on a virtual machine (guest machine), choose Tools | Vagrant | Halt on the main menu or run the following command in the embedded Terminal:

```
vagrant halt
```

For details, see http://docs.vagrantup.com/v2/cli/halt.html .

- To shut down the operating system on a virtual machine (guest machine), stop the virtual machine itself, and remove the resources provisioned on it during the creation (`vagrant up`), choose Tools | Vagrant | Destroy on the main menu or run the following command in the embedded Terminal:

```
vagrant destroy
```

For details, see http://docs.vagrantup.com/v2/cli/destroy.html .

Creating and Removing Vagrant Boxes

On this page:

- Introduction
- Creating a Vagrant box
- Deleting a Vagrant box

Introduction

It is possible to work with multiple virtual boxes, created in Vagrant. With PhpStorm, you can add new boxes without leaving the IDE.

The unnecessary boxes can be easily deleted .

Creating a Vagrant box

1. Open the Settings dialog box .
2. Under the Project Settings, click Vagrant .
3. Click + button, located next to the Vagrant Boxes table.
4. In the Add Vagrant Box dialog box, specify the name of the new box, and its URL. Then click OK.
   The new box is added to the list of available boxes.

Deleting a Vagrant box

1. In the Vagrant page of the project settings, under the Vagrant Boxes table, select the box to be deleted.
2. Click –, and confirm deletion.

The selected box is removed from the list.

Initializing Vagrant Boxes

On this page:

– Basics

– Initializing a virtual box

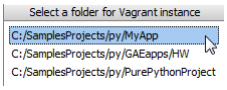
– Activating a virtual box

Basics

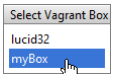
PhpStorm makes it possible to execute the `init` procedure on a project root. After performing `init` on a project root, the Vagrant configuration file named `Vagrantfile` is created there. This file can be used for creating a remote interpreter .

Initializing a virtual box

1. On the main menu, choose Tools | Vagrant | Init in Project Root.
2. In the Select Vagrant Folder pop-up menu, select the project root to be used:



3. Since it is possible to have multiple virtual boxes, specify the particular box you want to work with:



The `vagrant init` command is launched, and shows its output messages in the Run tool window.

If the Vagrant executable is not specified in the Vagrant page, you can still initialize a Vagrant box. However, in this situation, PhpStorm pops up a file browser, enabling you to select Vagrant executable.

Activating a virtual box

Once `init` is successfully completed, you are ready to perform the `vagrant up` command and import to the root the box used at initialization.

To activate a virtual box by executing the `Vagrant up` command, choose Tools | Vagrant | Up on the main menu. The `vagrant up` command is launched, and shows its output messages in the Run tool window.

XPath and XSLT Support

In this section:

- XPath and XSLT Support
  - Prerequisites
  - Overview
  - Changes to the UI
- XPath Expression Evaluation
- XPath Search
- XPath Expression Generation
- Plugin Settings
- XSLT Support

Prerequisites

Before you start working with XSLT, make sure that the XPath View+XSLT Support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

XSLT support is available in all XML files that declare the XSLT-Namespace `http://www.w3.org/1999/XSL/Transform` on their root element.

Overview

The plugin lets you do the following:

- Evaluate expressions against the currently focused document, including support for the `document()` function to make cross-document queries
- Evaluate an expression against multiple XML documents in a *Find in Path* style way by the new Find by XPath action.

The Expression Evaluation dialog allows you to highlight matching expressions in the current editor or to open the Find tool window to display a list of matching lines. Editing XPath expressions is enhanced by on-the-fly error-checking including a set of customizable XPath Inspections and a wide range of code completion suggestions.

The plugin also offers a way to display a unique XPath expression for a selected element in an editor. This is available via View | Unique XPath in the Main Menu.

XSLT support is not limited to XPath expressions, it also supports a wide range of XSLT constructs, like checking the existence of templates that are called via `xsl:call-template` and their parameters, refactoring and navigation enhancements. Find out more about this in the XSLT Support section.

Changes to the UI

- XPath viewer settings
- XSLT settings
- XSLT run/debug configuration

XPath Expression Evaluation

On this page:

- Overview
- Simple mode
- Advanced mode
- Options

Overview

Evaluating XPath expressions in PhpStorm has two main purposes: Testing XPath expressions that are to be used in program code or XSLT scripts, and making structured queries against XML documents. The Evaluate XPath action can be invoked either from the editor context menu, the main Search Menu or the main toolbar by clicking the action's icon .

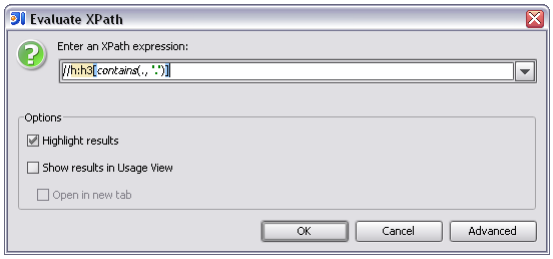
The Expression Evaluation has two different modes, a *simple* mode, that allows you to enter simple one-line expressions and doesn't allow you to configure and Context settings. For more convenient editing of long expressions in a multi-line style way, there's the *advanced* mode dialog that also has a button to edit some context settings, such as namespaces and their prefixes, and variables to use for the evaluation.

Both dialogs feature a history of the recently evaluated expressions, completion, syntax-checking and highlighting, as well as some semantic error checking of the entered expression. Semantic checks include validation of used namespace prefixes, useless XPath expressions (e.g. `@comment()`) and node tests for element/attribute names that don't occur in the context document and would not be successfully matched.

Some error checks and XPath inspections also provide Quick Fixes for detected problems, e.g. the possibility to map an unresolved namespace-prefix to a URI by intention.

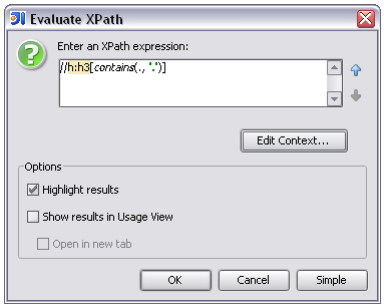
Simple mode

The simple mode comes with an input field that can be used to enter simple one-line expressions that don't require any customization of namespace prefixes or make use of predefined variables. The last recently used expressions can be selected from the drop-down list.



Advanced Mode

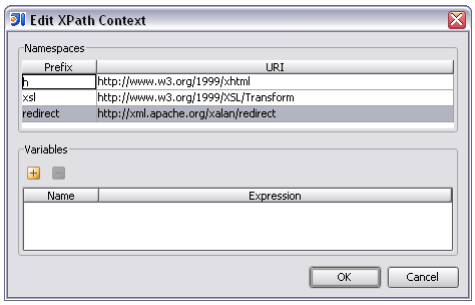
The advanced mode adds the possibility to edit expressions in a multiline editor and has another button to edit the XPath Context . The expression history can be browsed by the Up- and Down arrays on the right of the dialog or the keyboard shortcuts for the previous/next history element (usually Ctrl-Alt-Up, Ctrl-Alt-Down).



XPath Context

This dialog allows you to assign custom prefixes to the namespace URIs that are used in the context document. This can be useful to assign a shorter prefix, resolve prefix clashes or to actually define a prefix for the default namespace. This can be essential because XPath does not automatically match elements in the default namespace without specifying a prefix for the element to be matched.

The XPath Context also includes the possibility to define custom XPath variables that can be used in queries for repeating expressions. Each variable in the table can be assigned an expression that will be evaluated once when the query is executed. The resulting value is then available for multiple use at no additional computational cost.



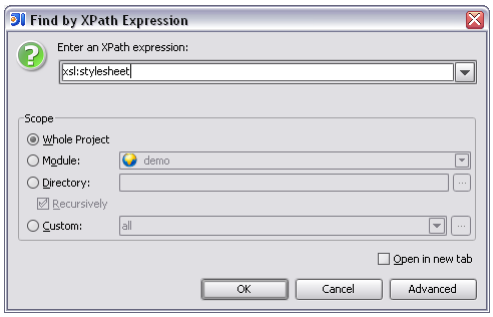
Options

- Highlight results: This option highlights the matched nodes in the current editor. Matched nodes that don't belong to the current editor (may happen by using the `document()` function) are not highlighted. It's recommended to display such cross-document results in the Find Usages toolwindow.

- Show results in Usage View: this option shows all matched nodes in the Find Usages toolwindow. Check Open in new tab to open the result in a new tab instead of reusing the last one.

XPath Search

This new feature is the XML-aware counterpart to PhpStorm built-in functions *Find in Path* and *Search Structurally*: It allows you to find occurrences of certain XPath expressions in all XML files in a specific scope. It is available as Find by XPath... under the Search menu.



The scope that can be chosen is either the full project scope, a specific module or a simple directory. There's also the ability to use *custom* scopes, but this is somewhat limited as it will only scan XML files inside source-folders.

This dialog provides the same functionality regarding completion, error checking and intentions as described in Evaluate Expression .

XPath Expression Generation

This action computes a unique XPath expression that matches the currently selected node in the document. The action is available from the View-Menu (Unique Path) and the Editor-Context Menu (Show Unique XPath). The action is only enabled when the caret is placed on an element that a useful expression can be generated for.

```
<xsl:param name="element" />
<xsl:pa
```

XPath: /xsl:stylesheet/xsl:param[@name='element']

> **Tip** The generated expression in the pop-up can be selected and copied into the clipboard for further use.

If a simple XPath expression like `/root/something/else` doesn't produce a unique result, the action has two strategies to make it unique:

- If the non-unique node is an element, the action looks for attributes with the name 'id', 'name' and attributes that are of ID-type, as defined by the document's DTD or XML Schema.
  Example: `/root/something[@id="foo"]/else`
- For nodes other than elements (comments, processing instructions), or if the above rule doesn't produce a unique result, the index of the node inside its parent is appended.
  Example: `/root/something/else[2]`

Plugin Settings

XPathView plugin settings are configured on the XPath Viewer page of the Settings .

Item	Description
Scroll first hit into visible area	Select this check box to have the editor automatically scroll to the first XPath match.
Use node at cursor as context node	Select this check box to have the entered XPath expression use the currently selected node (tag/attribute/pi, etc.) as its context node and evaluate the expression relatively to this node.
Highlight only start tag instead of whole tag content	Do one of the following: − Select this check box to have only the name of a matching tag highlighted. − Clear this check box to have the entire content of a matching tag highlighted.
Add error stripe markers for each result	Select this check box to have each match supplied with an error stripe marker which can be quickly navigated to. The tooltip of each marker shows the matched content.
Show actions in Toolbar	Select this check box to have buttons that invoke XPath-related actions displayed on the Main Toolbar .
Show actions in Main Menu	When this check box is selected, XPath-related actions are available from the main menu.
Colors	In this area, configure color indication during execution of XPath expressions. Clicking on the color box will open the Select Color dialog in which you can modify the current color indication. − Highlight Color - in this area, select the color to indicate XPath matches in the editor. − Context Node Color - in this area, select the color to indicate the current context node.

XSLT Support

- Completion
- Error Highlighting
- Documentation
- Refactoring
- File Associations

Completion

On this page:

- Overview
- Completion in XPath expressions
- Completion for template names in xsl:call-template
- Completion for template parameters

Overview

With Code Completion being one of the key features of PhpStorm, the plugin provides several possibilities to complete keywords, predefined functions, variables and parameters used in XPath expressions, template names and names of parameters that can be passed to a template invocation.

Completion in XPath Expressions

It's possible to complete all parameters/variables in scope inside an XPath expression in a normal expression attribute or inside an attribute value template

```
<xsl:message>
  <xsl:value-of select="concat('Param: ', $my-)" />
</xsl:message>
```

$my-param1
$my-param2
$my-unused-param

Also, all predefined functions and keywords are available for completion, including function signatures.

```
<xsl:value-of select="st" />
```

starts-with(string, string) boolean
string(any?) string
string-length(string?) number

The Quick Documentation Lookup also works in completion lookup lists.

Completion for Template Names in Xsl:Call-Template

The template's name that is to be called can be completed from a list of all named templates in the current document and included stylesheets

```
<xsl:call-template name="my-template">
  <xsl:with-param na
  <xsl:with-param na
```

my-template
my-template2

Completion for Template Parameters

There's a special completion for parameters that should be passed to a template in a `xsl:call-template` invocation. The completion lists all parameters that are declared by the template and are not yet present in the *argument list* of the invocation, i.e. there's no `xsl:with-param` yet.

```
<xsl:call-template name="my-template">
  <xsl:with-param name="my-param1" select="123" />
  <xsl:with-param name="my-"
  <xsl:with-param                    am" select="" />
</xsl:call-templat
```

my-param2
my-unused-param

Tip That's especially useful when also using completion to create the `xsl:with-param` tag, because this will automatically trigger the completion for the parameter's name which is required according to the schema; e.g.

`<xsl:with-p` <ctrl-space>

=>

`<xsl:with-param name="` [lookup list] `"`

Error Highlighting

The XSLT support can detect a range of errors in XSLT constructs, such as misspelled template names, missing template parameters, bad match-patterns, references to undeclared variables, wrong or useless embedded XPath expressions, etc. and also offers Quick Fixes to automatically fix some of those errors.

- Syntax highlighting
- XPath syntax checks
- XPath type checking
- Pattern validation
- Unresolved references
- Duplicate declarations
- Other checks

Syntax highlighting

Where allowed, XPath function-calls, Axis names, Numbers, Strings, etc. are highlighted according to the currently active color scheme. By default, the plugin uses the colors that are defined for the corresponding Java types, such as number- and string literals, etc. If a different coloring is desired, those colors can be configured in IDE-Settings | Colors & Fonts on the tab XPath.

XPath syntax checks

Just like the interactive XPath Expression Evaluation, the XSLT support catches any syntax error in XPath expressions used inside a stylesheet.

```
<xsl:value-of select="foo/bar/" />
                              location step expected
```

XPath type checking

In XPath, almost all types are assignable to each other with certain well-defined conversion semantics. However, the conversion to NODESET isn't defined for any type and there's also no (portable) conversion function available. Such type-conversions are highlighted as an error.

```
<xsl:value-of select="name(12345)" />
                              Expected type 'nodeset', got 'number'
```

Pattern validation

A special form of XPath expression are patterns in XSLT. They are e.g. used as the value of the *match*-attribute in `xsl:template` elements. Only a certain subset of XPath expressions is allowed here, which the XSLT support checks for.

```
<!-- Pattern validation -->
<xsl:template match="string()" />
                      Bad pattern
```

Unresolved references

References to variables that have not been declared or are not accessible from the current scope are detected and highlighted as an error. There are Quick Fixes available to create a variable or parameter declaration for such unresolved variables references.

```
<!-- undefined variable check -->
<xsl:message>
  <xsl:value-of select="concat('Variable: ', $my-variable)" />
</xsl:message>                                Unresolved variable 'my-variable'
```

Quick-fixes:

```
<xsl:message>
  <xsl:value-of select="concat('Variable: ', $my-variable)" />
  Create Variable 'my-variable'
  Create Parameter 'my-variable'
```

Duplicate declarations

In XSLT there must not be more than one variable or parameter declared on the same scope level. It's also not allowed that there is more than one template with the same name. The plugin will identify such duplicate declarations and highlight them in the editor.

Other checks

Shadowed variables

Even though is possible to have identically named variables or parameters in different nesting levels, this can be confusing and is likely to cause programming mistakes. The plugin can identify shadowed declarations and offers Quick-Fixes to either rename the local or the outer variable.

Missing template arguments

Another check that the XSLT support performs is whether all required parameters are specified with a `xsl:call-template`. A parameter is considered required if there is no default value, i.e. if there's no *select* attribute and the paremeter's declaring element has an empty body.

```
<xsl:call-template name="my-template">
                      Missing template parameter: my-param2
```

Quick-Fixes:

```
<xsl:call-template name="my-template">
```

Add Argument for 'my-param2'

Superfluous template arguments

There's also a supplemental check that flags arguments that are not declared as template parameters. There are Quick Fixes available to either remove the argument from the template-call or to add a corresponding parameter to the called template.

```
 <xsl:with-param name="my-undeclared-param" select="" />
/xsl:call-template>
```

Superfluous template parameter: my-undeclared-param

Quick-fixes:

```
<xsl:with-param name="my-undeclared-param" select="" />
```

Add Parameter 'my-undeclared-param' to Template 'my-template'

Remove Argument 'my-undeclared-param'

Function call arguments

The XSLT support does, just like the interactive XPath Expression Evaluation, check whether the number and types of function arguments match their declaration for built-in functions of XPath and XSLT.

```
<!-- missing function parameter check -->
<xsl:message select="concat($my-param2)" />
```

Function 'concat' requires 2 arguments

XPath inspections

All XPath Inspections are supported for editing XSLT documents. Those inspections can also be suppressed in a way that is similar to the standard suppression-mechanism PhpStorm uses for Java code by using *noinspection* XML comments. The suppression is possible on different levels, either on instruction level, template-level (if applicable) or stylesheet level.

XPath Inspections

The XPath language implementation provides a few built-in inspections that can check for common coding mistakes when writing XPath expressions both in the interactive mode and when writing XSLT scripts. Those inspection also provide a number of configuration options which can be configured on the Inspections page of the Settings dialog box.

Due to the way how these inspections are implemented and integrated, they only work for the on-the-fly editor highlighting and *not* if the inspections are run via Code | Inspect Code.

XPath Type Checking

There are two inspections that deal with type-conversion in XPath expressions: Implicit Type Conversion and Redundant Type Conversion.

Implicit Type Conversion

This inspection checks for any implicit conversions between the predefined XPath-types STRING, NUMBER, BOOLEAN and NODESET. While this is usually not a problem as the conversions are well-defined by the standard, this inspection can help to write XSLT scripts that are more expressive about types and can even help to avoid subtle bugs:

`<xsl:if test="foo" />` is not the same as `<xsl:if test="string(foo)" />`

The first test checks whether the element *foo* exists (`count(foo) > 0)` , the latter one however is only true if the element actually contains any text (`string-length(foo) > 0`). The plugin will then offer to make the type-conversion more explicit.

There are several options to adjust the inspection to personal preferences by offering the possibility to individually enable it for implicit conversions between certain types.

The plugin can also be told to always flag explicit conversions that do not result in the actually expected type, such as `<xsl:if test="number(foo)" />` and provides a special option to ignore the conversion from NODESET to BOOLEAN by using the `string()` function as a shortcut for writing `string-length() > 0` .

Redundant Type Conversion

This inspection checks whether any type-conversion with the functions `string()` , `number()` or `boolean()` is redundant, i.e. whether the type of argument is the same as the functions return type or if the expected type of the expression is of type *any*. While such an explicit conversion may sometimes be intentional to emphasize the type, this can usually be safely removed.

Expression validity checks

Those inspections check whether an expressions contains any potential semantic mistakes, such as referencing element/attribute names that don't occur in instance documents or using predicates that don't potentially match anything.

Check Node Test

This inspection checks whether any element/attribute names that are used in XPath-expressions are actually part of an associated XML file or are defined in a referenced schema. This helps to avoid problems caused by typos in XPath-expressions that would otherwise occur when running the script and may even then not be recognized immediately.

Example:

```
<xsl:template match="h:txtarea" />
```

If the prefix *h* is bound to the XHTML namespace, the inspection will flag this part of the match-expression as an unknown element name because the correct name of the element is *textarea*.

Index Zero Usage

This inspection checks for any accidental use of zero in a predicate index or in a comparison with the function `position()` . Such is almost always a bug because in XPath, the index starts at one, *not* at zero.

Example:

`//someelement[position() = 0]` or `//something[0]`

Developing custom XPath inspections

The XPath inspections make use of the normal inspections API of PhpStorm. However, due to the way the XPath Language-Support is integrated, this is a bit more complicated and it's at the moment not readily possible to develop full-blown 3rd-party XPath inspections. While it's theoretically possible to develop custom inspections that make use of the XPath-PSI API and are derived from `org.intellij.lang.xpath.validation.inspections.XPathInspection` , this is not recommended and not supported.

Please contact me if there's a need for a special inspection or there is significant interest that would justify the effort to make this more pluggable.

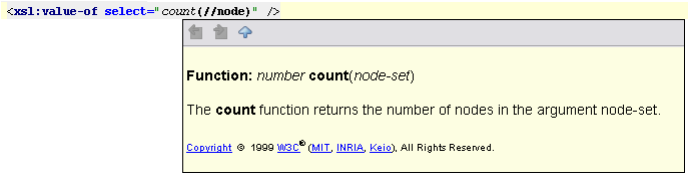
Documentation

On this page:

- Documentation for XSLT-elements and predefined XPath-functions
- Custom documentation

Documentation for XSLT-elements and predefined XPath-functions

The plugin comes with bundled documentation for XSLT elements and functions that is accessible via the Quick Documentation Lookup (Ctrl-Q) action. This documentation is extracted from the respective sections in the official W3C XSLT and XPath documents.



Custom documentation

It's also possible to annotate elements in an XSLT stylesheet with some documentation fragments. This can be looked up by the Quick Documentation Lookup function for variables, parameters and templates

Refactoring

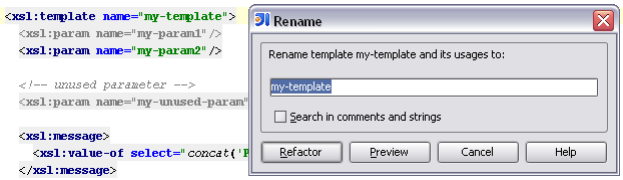
The plugin supports some of PhpStorm built-in refactorings Rename and Safe Delete for XSLT items such as templates, variables and parameters. It's also possible to create XSLT variables from selected XPath expressions with the well-known Introduce Variable shortcut. Inlining variables is possible with the Inline shortcut as well.

On this page:

- Renaming templates
- Rename variables and parameters
- Safe delete
- Introduce variable
- Introduce parameter
- Inline variable

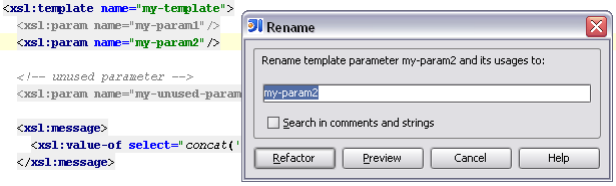
Renaming templates

Named templates can be renamed in PhpStorm just like any other symbol. All `xsl:call-template` invocations that refer to this template will be updated accordingly.



Rename variables and parameters

Just as named templates, it is possible to rename XSLT variables and template parameters, either at a point of their use or at their declaration.



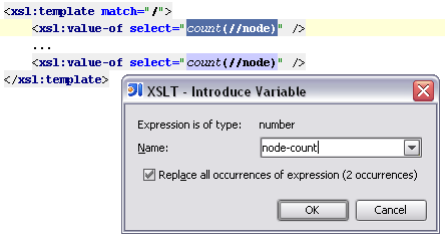
Safe delete

Named templates, parameters and variables can be deleted using PhpStorm Safe Delete feature, i.e. the item will be removed if there aren't any references left to it in other stylesheets across the project.

This is especially useful if the stylesheet may be included in other ones via `xsl:include` or `xsl:import` to make sure that nothing will be deleted that is still used somewhere else.

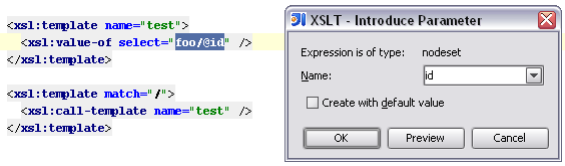
Introduce variable

It is possible to extract XPath-Expressions and turn them into an xsl:variable declaration. Check the Replace all occurrences check box to replace all other occurrences of the same expression.



Introduce parameter

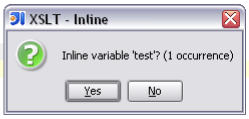
This is similar to Introduce Variable, but it creates a new parameter instead of a variable. It also has an additional option Create with default value that determines if the selected expression should be added as the introduced parameter's default value or if all calls to the template should be updated to pass the selected expression. That option is only available when introducing parameters to named templates.



Inline variable

This is just the opposite of Introduce Variable, it replaces all usages of a variable with the expression that is specified in the variable's select-attribute. Variables that don't have such an attribute cannot be inlined. Inlining variable references that resolve to parameters is not possible as well.

```
<xsl:variable name="test" select="
<xsl:value-of select="$test" />
```



File Associations

On this page:

- Overview
- Managing associations from the editor
- Removing an association

Overview

File Associations are used to associate an XSLT file with other XML files. This is currently used for three purposes:

- Enhanced completion for element- and attribute names in XSLT node-selections. The completion will offer all element- and tag names that are found in the associated documents.
- Enhanced error highlighting for XSLT node-selections. If an XSLT script has been associated with one or more XML files, any references to element- and attribute names that are not part of the associated files fill be flagged with the warning message: " Tag name '...' is not part of the document".
- File Associations are also used for creating Run Configurations. The XML input file to use for the transformation can be conveniently chosen from the list of associated files.

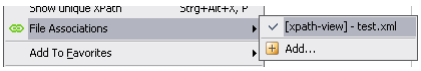
XSLT File Associations are defined per project and managed in the XSLT File Associations settings page.

Managing associations from the editor

Associations can be also created by invoking the Add... action from the File Associations group in the Editor Context Menu. A file-selection dialog opens that can be used to select one or more X(HT)ML files to associate the XSLT file with.

Removing an association

An Association can be removed by clicking on the corresponding file name in the File Associations group. The file names are displayed with a path that is relative from the current file. If the associated file is part of any module, the module name is included in square brackets.



It's also possible to invoke the associations configuration dialog through the Configure... action. This opens the associations configuration dialog and preselects the file that's currently opened in the editor.

Tutorials

- Configuring Generic Task Server
- Finding and Replacing Text in File Using Regular Expressions
- TODO Example
- Using Live Templates in TODO Comments
- Using Emacs as an External Editor
- Using PhpStorm as the Vim Editor

Configuring Generic Task Server

On this page:

- Introduction
- Prerequisites
- Specifying the task server
- Configuring the task server
- Choosing the response type and using selectors
- Obtaining issues from the server
- Summary

Introduction

The Generic connector has been developed as a mechanism enabling the users to configure integration with the servers that PhpStorm does not officially support. At the moment, this mechanism:

- Supports the services with REST API.
- Does not support pagination in server responses.
- Allows the following ways of client authentication:
  - Basic HTTP authentication
  - Sending a preliminary request to the specified address (which is required, for example, by YouTrack).
- Supports GET and POST requests. At this moment POST requests can contain only form-encoded data , specified as query parameters in the corresponding URL

Note that a client can only send requests to a server and retrieve responses according to a certain template.

This tutorial is created for the JIRA 5.0 server. However, configuring a generic server described below is valid for any higher version of JIRA, including JIRA OnDemand.

Prerequisites

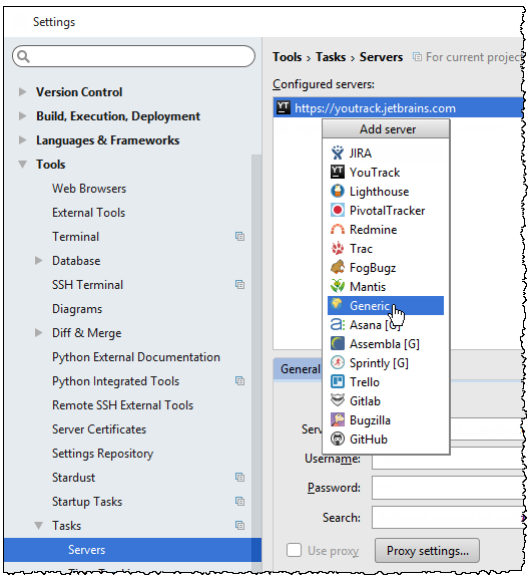
Make sure that the following prerequisites are met:

- You are working with PhpStorm version 8 or later.
  If you still do not have PhpStorm, download it from this page . To install PhpStorm, follow the instructions , depending on your platform.

  This tutorial is created with PhpStorm version 2016.2.
- You have an account in a tracking system. For example, this tutorial makes use of JIRA.

Specifying the task server

1. Open the Settings / Preferences Dialog by pressing `Ctrl+Alt+S` or by choosing File | Settings for Windows and Linux or PhpStorm | Preferences for OS X. Expand the Tools node, and then click Servers under Tasks.
2. On the Servers page that opens, click + and choose Generic:



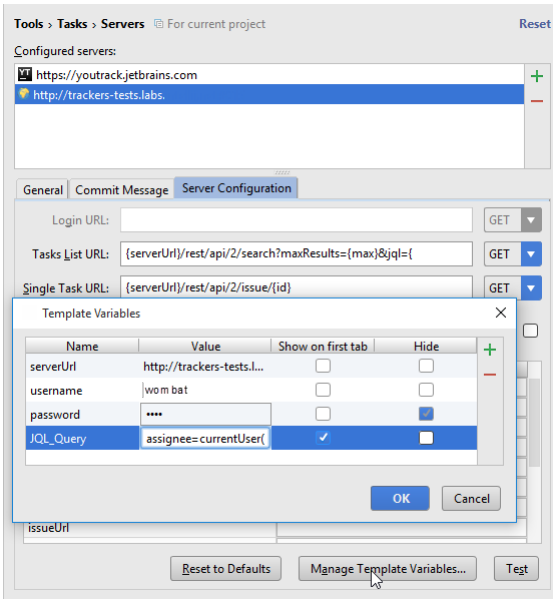
3. In the General tab, specify URL of the server, select the check box Use HTTP Authentication, and type your user name and password.
4. Leave the Commit Message tab as is, and then click the Server Configuration tab.

Configuring the task server

The Server Configuration tab contains three fields:

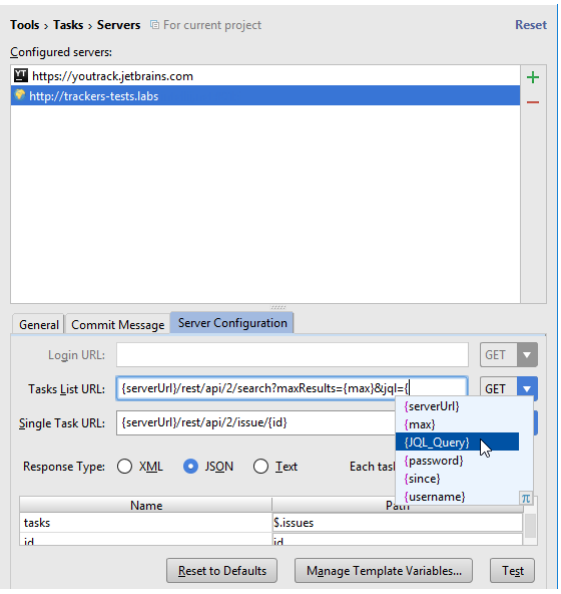
- The Login URL field is disabled, because the check box Use HTTP authentication in the General tab has been selected.
- The Tasks list URL field makes it possible to obtain the list of issues from the server (this list is asked for when you open a task).
  In this field, specify the variable `{serverUrl}` that corresponds to the server URL, specified in the General tab, then type the path `/rest/api/2/search` and expression `?maxResults={max}&jql={JQL_Query}` .

  Pay attention to the template variable `{JQL_Query}` , which is not defined so far. To define it, click the button Manage Template Variables, then in the Template Variables dialog box that opens, click + and type the name and value of this template variable:



Warning! This is important! For an undefined template variable, the test connection will fail!

Defining a template variable leads to including it in the suggestion list:



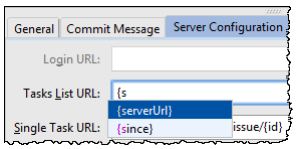
Mind the check box Show on first tab in the Template Variables dialog box. If this check box is selected (as it's done in this tutorial), the template variable and its value shows on the General tab of the Servers page.

Note It is also possible to use an absolute path in the format `http://<server URL>:<port>/rest/api/2/search<expression>`.

- The Single task URL field makes it possible to obtain a detailed information about a specific issue by its ID. This field is optional, when the check box Each task in separate request is cleared.
  To fill in the field, use the following format

  <variable `{serverUrl}` that corresponds to the server URL, specified in the General tab> <the path `/rest/api/2/issue/` > <variable `{id}` >

Code completion is available in all these fields.



Choosing the response type and using selectors

You also need to choose the response type and to specify how information about an issue will be extracted from the server response. This is done in the table of selectors in the lower part of the Server Configuration tab.

PhpStorm suggests three response types: XML, JSON and text.

- Choose XML if the server responds in the XML format, the selectors are described in XPath.
- Choose JSON if the server responds in the JSON format, the selectors are described in JSONPath.
- Choose Text if the server responds in the plain text format, the selectors are described as regular expressions.

In this tutorial, for example, let's choose JSON and describe the selectors using JSONPath .

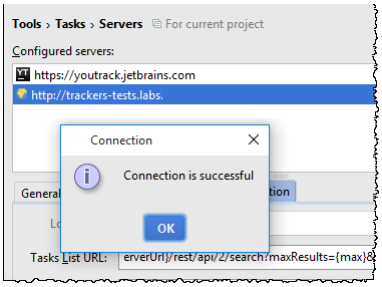
The first three selectors are mandatory:

- tasks: this selector denotes the path in the server response to the descriptions of specific issues.
- id: this selector denotes the path to a unique ID in the description of a specific issue.
- summary: this selector denotes the path to a title of an issue within its description.

To fill out the table, use the server responses. For JIRA, for example, one can observe the server responses immediately in the browser, following a link in the format `http://<server URL>:<port>/rest/api/2/search` .

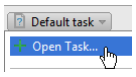
Obtaining issues from the server

1. To check that the specified settings ensure successful integration with the tracking system, click Test:

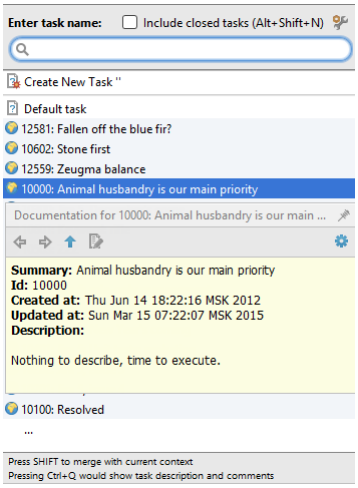


   If the connection is successful, click OK and close the Settings dialog.

2. Do one of the following:
   - On the main menu, choose Tools | Tasks & Contexts | Open Task
   - Press `Shift+Alt+N`.
   - Click the tasks combo on the main toolbar:



3. In the list of tasks, choose the one you want to open, and hit `Enter`:



Summary

In this tutorial, we have:

- assigned JIRA as the generic task server.
- configured this task server.
- specified response type and selectors.
- made sure that connection is successful.
- retrieved issues from the configured server.

Finding and Replacing Text in File Using Regular Expressions

On this page:

- Example code
- Finding and replacing a string using regular expression
- Changing case of the characters

Example code

Consider the following XML code fragment:

```
<new product="ij"
                    category="105" title="Multiline search and replace in the current
                    file" />
                    <new product="ij" category="105" title="Improved search and replace in the current
                    file" />
                    <new product="ij" category="105" title="Regexp shows replacement preview"
                    />
                    ...
```

Finding and replacing a string using regular expression

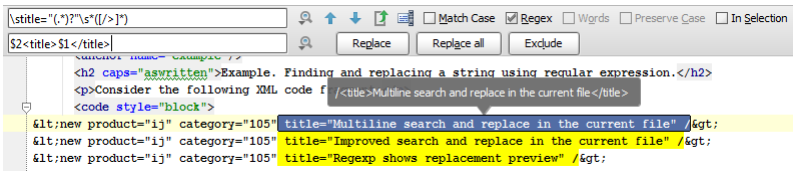
Suppose you want to replace an attribute within an element (`title`) with an expanded tag `<title></title>` , which contains some arbitrary string in double-quotes within.

This is how it's done.

1. With the XML file in question opened in the editor, press `Ctrl+R`. The Replace pane appears on top of the editor.
2. Since you want to replace all the `title` attributes, regardless of the actual strings contained therein, use regular expressions. Make sure that the check box Regex is selected. Thus everything you type in the Search and Replace fields will be perceived as the regular expressions.
3. In the Search field, start typing regular expression that describes all `title` attributes. Note that though the regular expression `\stitle=".*?"\s*[/>]*` matches contents of the `title` attribute, it is recommended to capture the groups for referencing them in the Replace field:

   ```
   \stitle="(.*)?"\s*(/>*)
   ```

   Note that for the regular expressions replacement preview is shown at the tooltip:

4. Then, in the Replace field, type the following regular expression:

   ```
   $2<title>$1</title>
   ```

5. Click Replace, or Replace All.

As you see, the second capture group (`/>`) is moved ahead to close the <new> element, while the first capture group `<` , which matches any string in double quotes, is moved to the element `<title>` .

Changing case of the characters

Suppose now that you want to change characters os the search strings. Again make sure that Regex check box is selected.

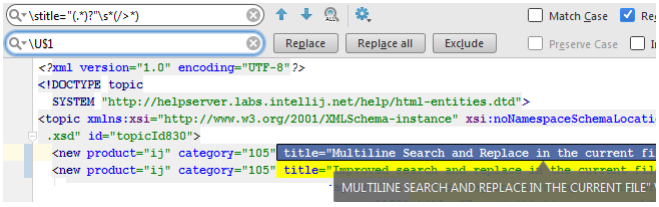
In the Search field, type the search expression:

```
\stitle="(.*)?"\s*(/>*)
```

Next, fill in the Replace field with the following expression:

```
\U$1
```

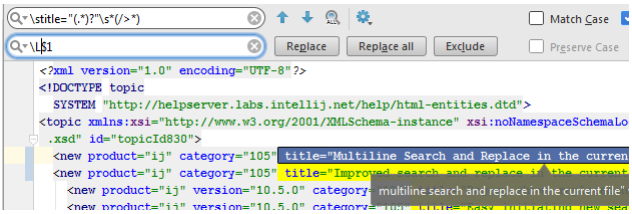
The found occurrences are replaced with the upper-case characters:



Next, let's make the strings all lower-case. To replace occurrences with the lower-case characters, type the following replacement string:

```
\L$1
```

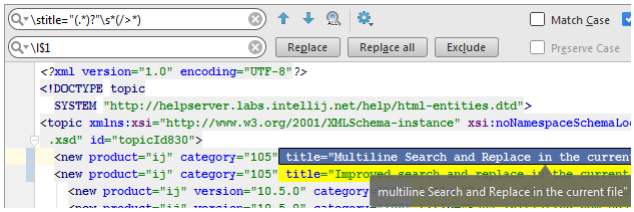
Then the suggested replacement will be:



And finally, if you want change the case of the first letter only, type the following replacement string:

```
\l$1
```

PhpStorm suggests the following replacement:

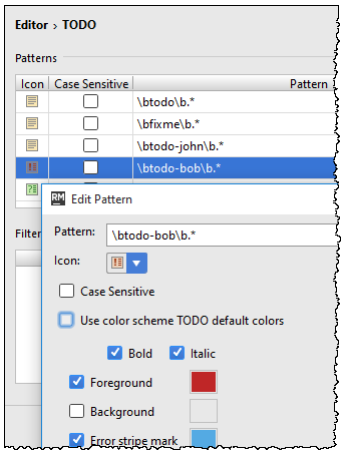


TODO Example

Consider the following example: creating and viewing TODO items for each of the team members.

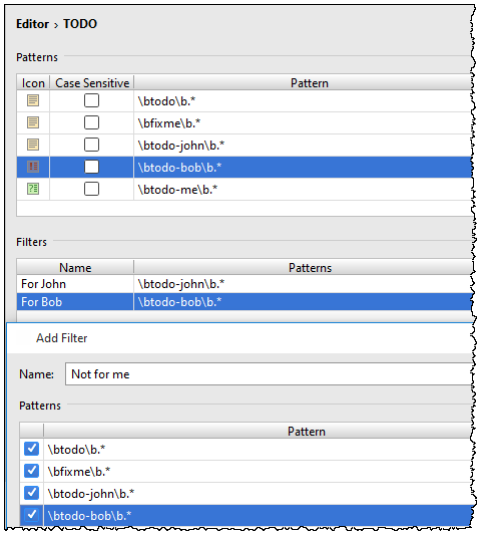
Creating patterns for TODO items

On the TODO page of the Settings dialog, click + in the Patterns section, and create additional TODO patterns , for example, `todo-John` , `todo-Bob` and `todo-me` , with new icons, and custom color schemes:



Creating filters

Next, create several filters, which you will use to show the TODO items, say, for each of the developers, and not for your good self. For this purpose, in the Filters section, click +, and specify the filter names, for example, `For John` , `For Bob` , and `not for me` . Associate these filters with the patterns:



Creating TODO items in source code

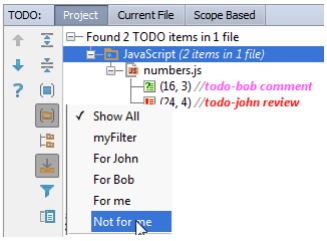
Now, in the source code, create TODO items: in the line of code, where you want to add a note, press `Ctrl+Slash` , or `Ctrl+Shift+Slash` , and type `TODO` that matches one of the patterns, followed by some meaningful description:

```
    //TODO-bob region Description
    math = {
        root: Math.sqrt,
        square: square,
        cube: function(x) {
            return x * square(x);
        }
    };
    race = function() {
        var runners, winner;
//      todo-me have a break

        //endregion
//TODO-john fix
        winner = arguments[0], runners = 2
```

Viewing TODO items

Having produced a number of TODO items across the whole project, review them in the TODO tool window: click the TODO button on the tool window bar to show the tool window. By default, all the encountered TODO items are displayed.

Let's now show the TODO items for Bob and John, and hide the other items: click the filter icon on the toolbar of the TODO tool window, and select Not for me in the menu:


TODO:
Project
Current File
Scope Based
Found 2 TODO items in 1 file
JavaScript (2 items in 1 file)
numbers.js
(16, 3) //todo-bob comment
(74, 4) //todo-john review
Show All
myFilter
For John
For Bob
For me
Not for me

Using Live Templates in TODO Comments

On this page:

- Overview
- Creating TODO pattern and filter
- Creating live template and variables
- Using the REVIEW items

Overview

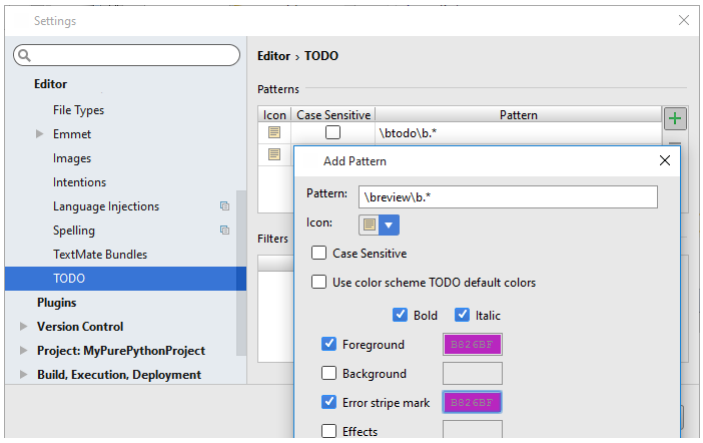
Let's explore an advanced PhpStorm's facility to create a live template for the TODO items' text. Why do we need it at all? For example, you want your team mates to create unified TODO items, with the user name automatically filled in, followed by some arbitrary text.

This is how it's done.

Creating TODO pattern and filter

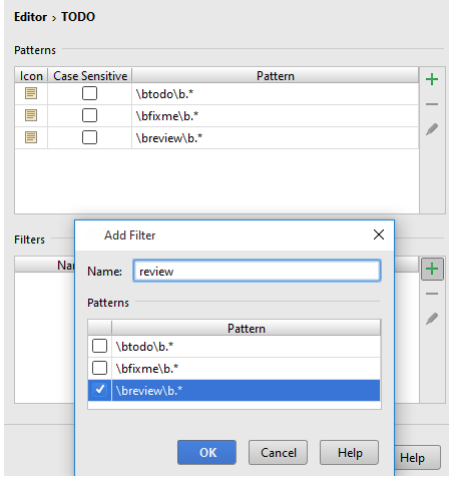
Open the Settings/Preferences dialog, and under the Editor section, click TODO .

Create pattern `review` . To do that, click + in the Patterns section:



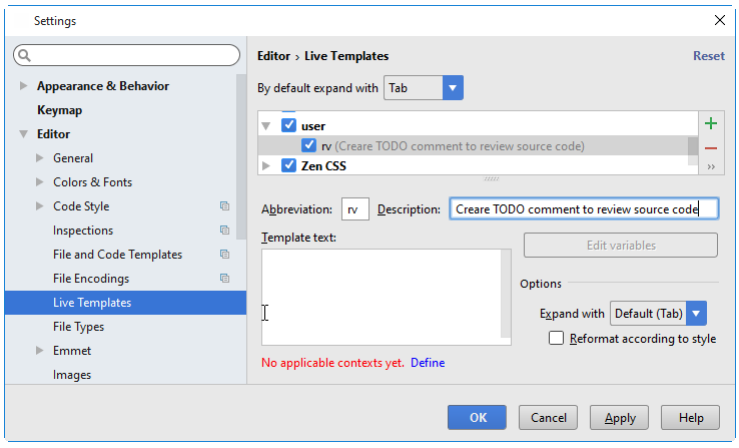
Define color in the Color Picker - in this case, it's pink.

Next, let's create a filter . To do that, click + in the Filters section, and define the filter:



Creating live template and variables

Next, back in the Settings/Preferences dialog, under the Editor section, click Live Templates .

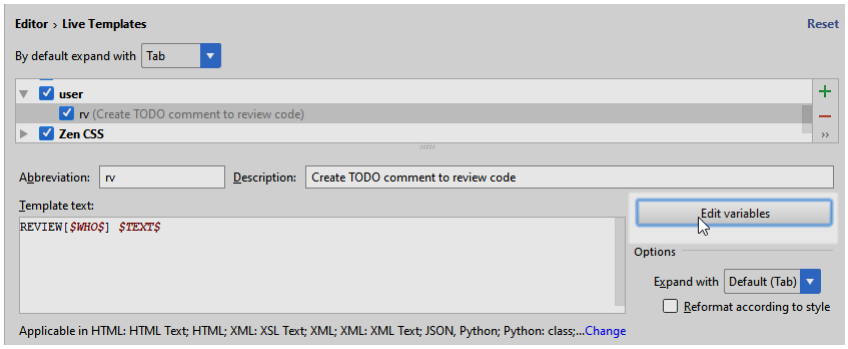


Note that the new template is added to the automatically created group `user`.

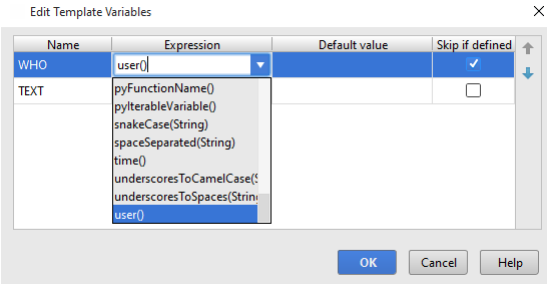
Next, pay attention to the red note at the bottom. It says that the new template lacks context, where it should apply. So let's click the link Define, and allow all possible contexts. And finally, let's define the body of the template itself: in the area Template text, type the following:

```
REVIEW[$WHO$] $TEXT$
```

We have two undefined variables here: `$WHO$` and `$TEXT$`. The variable `$TEXT$` will be used just as an input field, while the variable `$WHO$` should be filled in automatically. To define this variable, click the button Edit variables:



Next, in the Edit Template Variables dialog box, select an expression for the variable `$WHO$`:



Using the REVIEW items

Now let's make sure it works. Back in the editor, create a line comment (`Ctrl+Slash`), type `rv`, and press `TAB`:

```
# REVIEW[wombat]
```

REVIEW[wombat]

Note that a right-gutter stripe next to the TODO comment is also added to the editor. As you see, the live template `rv` has automatically populated the user name, leaving us with the task of just entering some meaningful comment:

```
# REVIEW[wombat] please make sure that everything works
```

Now, when you opt to show REVIEW comments only, use the filter. To do that, click and select the filter review to show those TODO comments only, that have the keyword REVIEW.

Using Emacs as an External Editor

On this page:

- What this tutorial is about
- Prerequisites
- Configuring Emacs as an external tool
- Opening current file in Emacs
- Assigning a keyboard shortcut

What this tutorial is about

This short tutorial aims to walk you step by step through defining Emacs as an external editor for PhpStorm.

Basics of Emacs are out of scope of this tutorial.

Prerequisites

Make sure that:

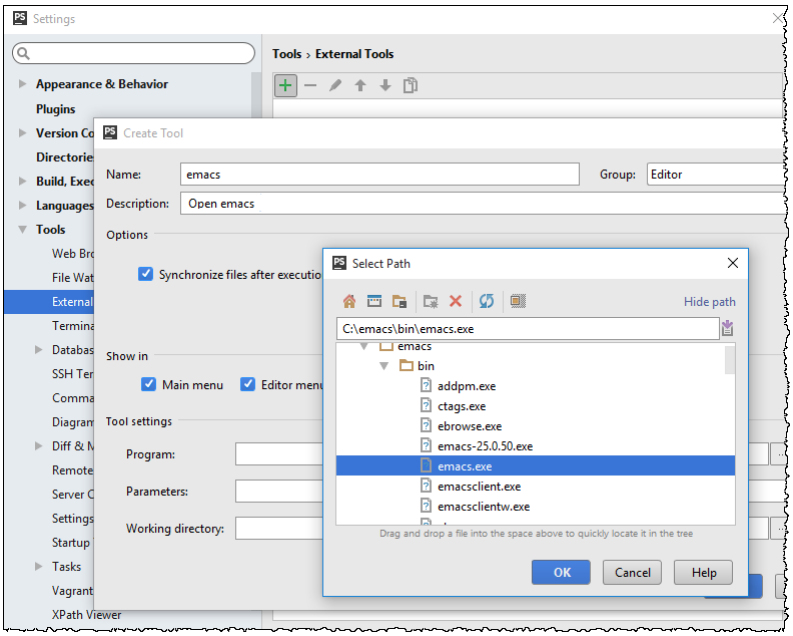
- Emacs is downloaded and properly installed on your computer.

Configuring Emacs as an external tool

Open Settings/Preferences dialog. To do that, you can, for example, choose File | Settings (on Windows and *nix) or PhpStorm| Preferences (on Mac OS), or click button on the main toolbar.

Then, under the Tools node, open the page External Tools . On this page, you have to specify your Emacs installation as an external editor for the current file. This is how it's done...

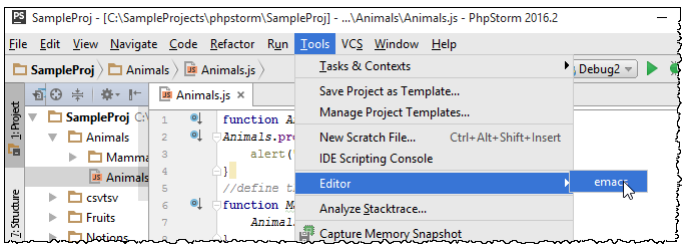
1. First, in the External Tools page, click +. The Create/Edit Tool dialog box opens.
2. In this dialog, do the following:
   - Type the tool name (Emacs) and optional description (Open Emacs)
   - Specify the name of the group, under which Emacs will appear in the Tools menu. In this example, the group name is Editors. This step is optional - if you specify no group name, then Emacs will appear in the Tools menu as is.
   - Clear the check box Open console.
   - Specify Emacs binary file location. You can either type it manually, or click the ellipsis button and find the desired binary in your file system.
   - Since you want to open the current file in Emacs, pass the file path as a parameter to the program: in the Parameters field, type `$FilePath$` .
   - Finally, specify the working directory - in our example, this is `$ProjectFileDir$`
   - Click OK.



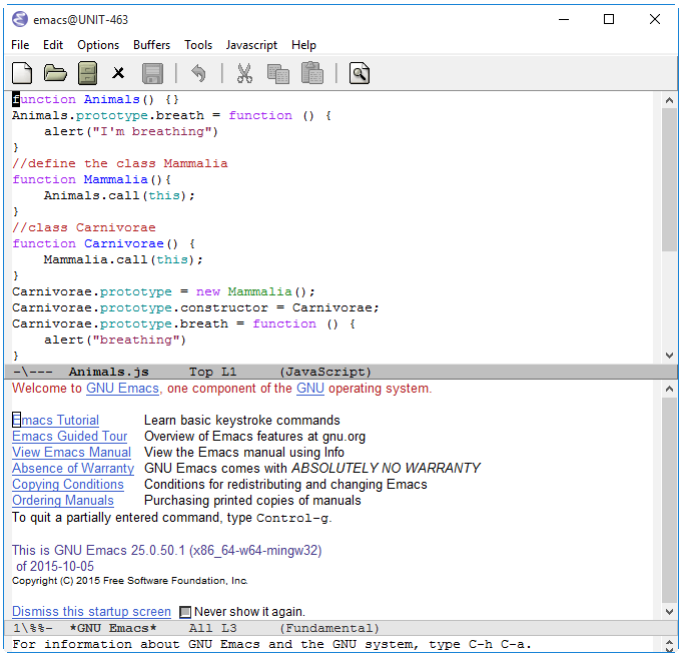
3. Apply changes and close the Settings/Preferences dialog.

Opening current file in Emacs

When you now look at the Tools menu, you will see the new node Editors. Pointing to this node reveals the Emacs command:

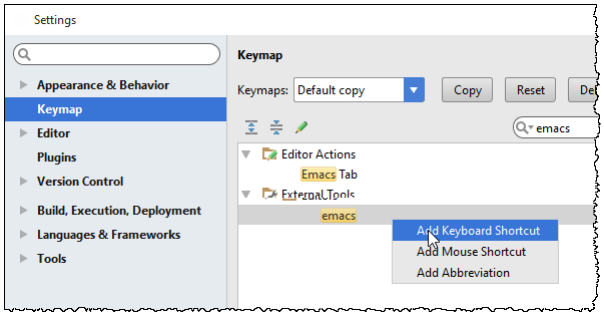


Open a file for editing . Next, on the Tools menu, choose Editors|Emacs - and see the current file in Emacs also:

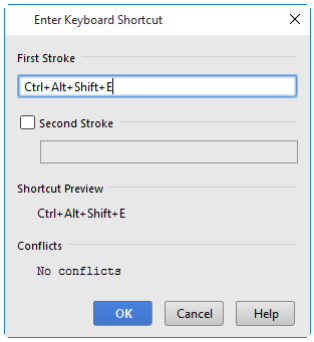


Assigning a keyboard shortcut

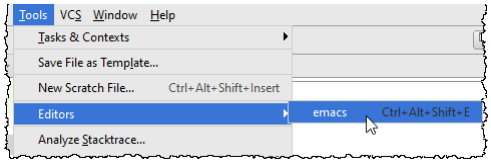
By the way, PhpStorm makes it possible to assign a keyboard shortcut to this action: click to open Settings/Preferences dialog, open the Keymap page , find Emacs, and choose Add Keyboard Shortcut on the context menu:



Enter Keyboard Shortcut Dialog dialog box opens, where you have to specify, which shortcut you would like this action to be associated with. Let's, for example, use `Ctrl+Shift+Alt+E`:



No conflicts are reported, so click OK, and see the new shortcut appearing in the list of actions and on the Tools | Editors | Emacs menu:



Using PhpStorm as the Vim Editor

On this page:

- Before you start
- Downloading and installing IdeaVim plugin
- What happens to PhpStorm's UI after restart?
- Configuring shortcuts
- Editing modes

Before you start

Make sure that:

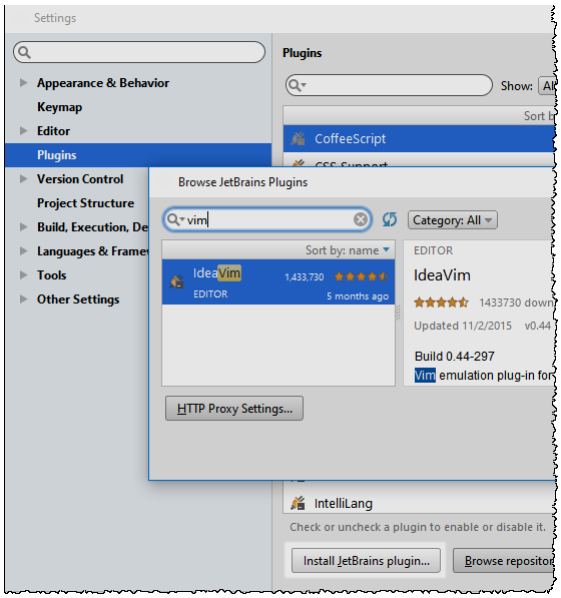
- You are working with PhpStorm version 2016.1 or higher. If you still do not have PhpStorm, download it from this page . To install PhpStorm, follow the instructions, depending on your platform.

  This tutorial is created with PhpStorm version 2016.1.

Downloading and installing IdeaVim plugin

On the toolbar of the PhpStorm main window, press `Ctrl+Alt+S` to open the Settings/Preferences dialog, and then click Plugins .

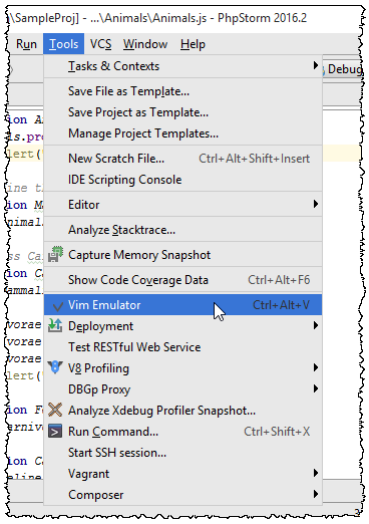
You see the list of plugins currently installed on you computer. However, the IdeaVim plugin is not among them. Click the button Browse JetBrains plugins. PhpStorm shows the contents of the huge JetBrains repository... you can type the word "vim" in the search field to narrow down the list:



After installing the plugin, it actually becomes available after PhpStorm restart.

What happens to PhpStorm's UI after restart?

First, on the Tools menu, a check command Vim Emulator appears:



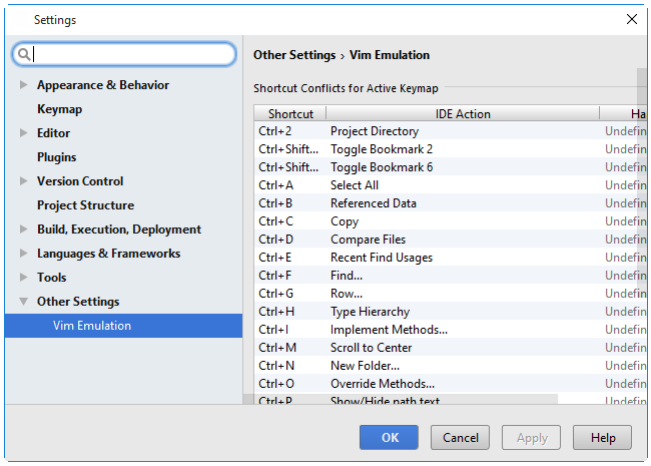
After PhpStorm restart, this check command is selected. You can disable Vim by clearing this check command.

Second, in Settings/Preferences dialog, an additional node Other Settings appears, with the page Vim Emulation. This page appears after restart!

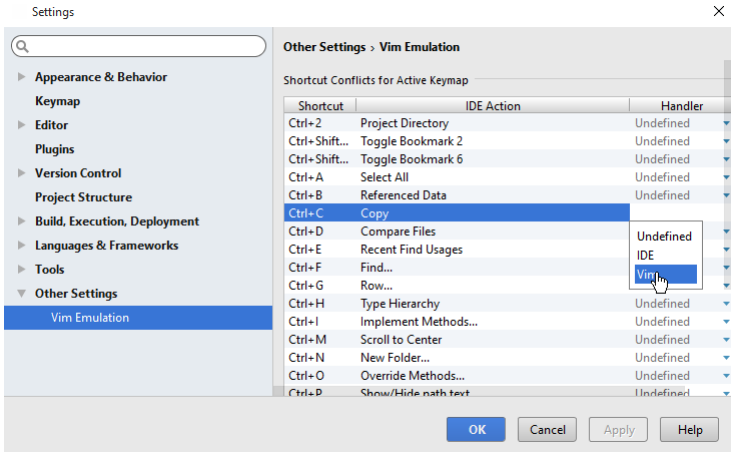
Configuring shortcuts

Both Vim and PhpStorm are keyboard-centric. With IdeaVim plugin, it is quite possible that PhpStorm's keymap runs into a conflict with the Vim keymap. That's why PhpStorm allows you choosing which keyboard shortcut you prefer for a certain action. This is how it's done.

Open Settings/Preferences dialog, and under the node Other Settings, click Vim Emulation:

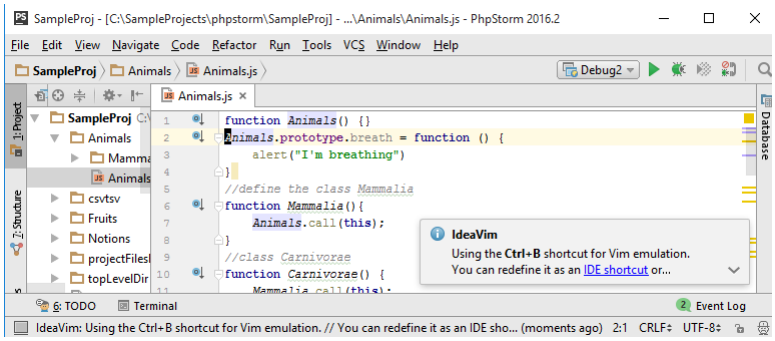


In the Shortcut column, select the shortcut you want to configure. Next, in the Handler column, click the corresponding cell, and see the drop-down list of three possible options (Undefined, Vim, IDE):



If you choose IDE, it means that the PhpStorm's shortcut for this particular action is enabled. When you press, say, `Ctrl+Z`, PhpStorm silently performs its action.

If you leave the handler undefined, then, on pressing the shortcut, say, `Ctrl+B`, PhpStorm shows the following banner:



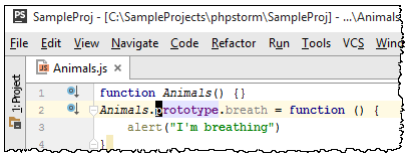
You can choose to redefine this shortcut as an IDE shortcut and thus accept the PhpStorm's keymap. To do so, click the link IDE shortcut.

If you click the link Vim Emulation, then PhpStorm will show the Vim Emulation page of the Settings/Preferences dialog.

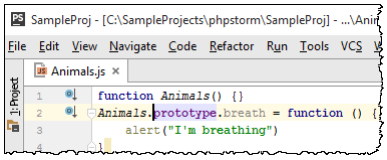
For the purposes of this tutorial, click the link Vim Emulation. Then, when you press `Ctrl+B`, PhpStorm will perform the Vim action for this keyboard shortcut.

Editing modes

OK, now that you have Vim enabled, you see that the cursor has changed its shape - now it is a block, which means that you are in the Normal mode :



If you want to enter the Insert mode , press `i`, and the cursor will turn into a line:



In this mode you can type new or change the existing code. Same way, you can enter the various Vim modes: for example, press `r` for the Replace mode .

By the way, as soon as you enter Vim emulation, it is also reported in the Status bar.

To return to the Normal mode, press `Escape`.

Reference

This part contains miscellaneous information related to the UI of the dialogs, tool windows and views, keyboard shortcuts, syntax references etc.:

- Essentials
- Dialogs
- Settings / Preferences Dialog
- Keyboard Shortcuts and Mouse Reference
- Tool Windows Reference
- Version Control Reference
- Table Editor
- Diagram Reference
- Icons Reference
- Regular Expression Syntax Reference
- Scope Language Syntax Reference
- Importing PhpStorm Settings on First Launch
- Project and IDE Settings
- Directories Used by PhpStorm to Store Settings, Caches, Plugins and Logs
- Managing Projects Under Version Control
- Color-Deficiency Adjustment
- Index of Menu Items
- Tuning PhpStorm

Essentials

PhpStorm is an advanced IDE. To get the most out of its capabilities and features, you should be familiar with its concepts. Concepts describe the basic notions of the IDE.

In this part:

- Project
- Content Root
- Scope
- Library
- Path Variables
- Supported Languages
- Running and Debugging
- Code Analysis
- Encoding

Project

On this page:

- Basics
- Project files
- Project types

Basics

Whatever you do in PhpStorm, you do that in the context of a project. A project is an organizational unit that represents a complete software solution. It serves as a basis for coding assistance, bulk refactoring, coding style consistency, etc.

PhpStorm does not support direct editing of files on remote hosts. Therefore to work with remote sources in PhpStorm, you need to download them and arrange them in a PhpStorm project . To keep your local and remote sources synchronized, configure automatic upload using the Upload changed files automatically to the default server drop-down list in the Options dialog box.

Project files

A project in PhpStorm is represented in the Directory Based Format. A project directory is marked with icon.

Such project directory contains the `.idea` directory, with the following files:

- `*.iml` file that describes the project structure.
- `workspace.xml` file that contains your workspace preferences.
- A number of `.xml` files. Each `.xml` file is responsible for its own set of settings, that can be recognized by its name: `projectCodeStyle.xml` , `encodings.xml` , `vcs.xml` etc.
  Thus, for example, adding a new run/debug configuration and changing encoding will affect two different `.xml` files. This helps avoid merge conflicts when the project settings are stored in a version control system and modified by the different team members.

All the settings files in the `.idea` directory should be put under version control except the `workspace.xml` , which stores your local preferences. The `workspace.xml` file should be marked as ignored by VCS .

`.idea` directory is not visible in the Project view of the Project tool window .

Project types

The directory structure of each project contains the `.idea` directory for the PhpStorm-specific settings and the project file, and libraries.

PhpStorm suggests the following types of projects:

- Empty project is intended for pure web programming. The directory structure of such project contains the `.idea` directory for the PhpStorm-specific settings and the project file, and libraries.
  Create a plain web project as described in the section Creating New Project from Scratch .
- Google App Engine for PHP project. This project type provides specific infrastructure of the Google App Engine for PHP application, and all the necessary files and settings.
  Create project as described in the section Preparing to Develop a Google App for PHP Application .

  > **Warning!** Python and Google App Engine SDK must be installed on your machine.

- Template-based projects are intended for developing web applications using third-party frameworks. PhpStorm creates specific infrastructure, with the files and directories structured in accordance with the requirements of the selected framework.
  - For client-side development, choose HTML5 Boilerplate , Twitter Bootstrap , Foundation , or Web Start Kit .
  - For server-side development, choose Node.js Express App .
  - Meteor App for both server-side and client-side development.
  - PhoneGap/Cordova App for developing mobile applications intended for running on various mobile platforms.
  - Web Start Kit .
  - Dart applications. Generating Dart application stubs requires that Dart is supported in PhpStorm:
    1. The Dart SDK is available on your computer.
    2. The Dart plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
  - Yeoman project generator, see Creating a Project Using Yeoman Generator .

Complete project creation as described in the section Generating a Project from a Framework Template .

Content Root

On this page:

- Basics
- Content root types

Basics

In the terms of PhpStorm, content is a collection of files with which you are currently working, possibly organized in a hierarchy of subfolders. The folder which is the highest in this hierarchy is called content root folder or content root (shown as) for short. A project has at least one content root folder, by default it is the project folder itself.

Having several content roots enables you to work with files from several directories that do not have a common immediate parent. This is helpful when you use some static contents, for example, icons. You can just save them all in a folder and then specify this folder as an extra content root in several projects.

Content root types

By default, all the files in a content root folder are involved in indexing, searching, parsing, code completion, etc. To change this status, folders within a content root can be assigned to the following categories:

- Source roots (or source folders; shown as).
  This is the root for namespaces used in your project.

  Based on this setting, PhpStorm suggests you the proper folder name when you want to create a new namespace under another parent namespace during creation or moving a PHP class, that is, when you are actually creating or moving a PHP class to a non-existing namespace under another parent namespace. If no Sources folder is specified, you will have to type the proper folder manually.

  Appointing a Sources folder is not mandatory but this helps you keep your project structure in compliance with the PSR0 and PSR4 standards. See Configuring PHP Namespaces in a Project for details.
- Test source roots (or test source folders; shown as).
- Resource roots (or resource folders; marked as).
  By assigning a folder to this category, you tell PhpStorm that files in it and in its subfolders can be referenced relative to this folder instead of specifying full paths to them.
- Excluded roots (shown as) are ones that PhpStorm "almost ignores".
  These roots contain files and folders ignored by PhpStorm when indexing, searching, parsing, watching etc.

  Excluded roots are not visible to PhpStorm. Usually, one would like to exclude temporary build folders, generated output, logs, and other project output. Excluding the unnecessary paths is a good way to significantly improve performance.

Scope

On this page:

- Basics
- Types of scopes
- Defining scopes
- Scopes coloring
- Predefined scopes

Basics

A scope is a subset of files, and/or directories in your project, to which you can limit the application of specific operations, e.g. search , code inspection , etc. Besides, you can configure coloring for each scope to see at once what sort of file you are dealing with.

Scopes get more helpful as your project grows larger. There are a lot of predefined scopes that cover basic cases. Additionally, it is possible to add custom scopes to your project. For example, you can create custom scopes for tests or for the files you are responsible for in your team.

Types of scopes

Scopes can be either shared or local:

- Shared scopes are accessible for the team members via VCS and are stored at the project level. in the `scopes` directory under `.idea` , as a file with the extension `xml` , i.e. `.idea/scopes/<scope_name>.xml` .
- Local scopes are intended for personal use only and are stored in your workspace in the file `workspace.xml` under `.idea` .

If necessary, you can share a local scope, make a shared scope local, or create a copy of the scope. For more information, see Configuring Scopes and File Colors .

Defining scopes

PhpStorm provides a special language that enables you to flexibly define the sets of entities included in a scope. See Scope Language Syntax Reference for details.

To create and edit scopes, use the Scopes page of the Settings dialog. Scopes are defined in the following modes:

- Manually, by specifying file masks according to the scope language syntax in the Pattern text box.
- By selecting files and folders and clicking the buttons Include, Include Recursively, Exclude, and Exclude Recursively. Based on the inclusion/exclusion, PhpStorm creates an expression and displays it in the Pattern.
  Refer to the section Configuring Scopes and File Colors .

You can explore the available scopes in the Project tool window .

Scopes coloring

Files belonging to different scopes can be highlighted in different colors throughout the PhpStorm's user interface: in navigation lists , in the editor tabs, in the Project window . This allows much faster and easier navigation in large projects.

If some file is included into several scopes, the order of the scopes becomes important: PhpStorm uses the color of the uppermost scope (shown in the Scopes settings page) to highlight such file. Of course, you can change the order of the scopes, and thus the resulted highlighting.

For detailed instructions on how to configure the scope order and scope-color associations, see Configuring Scopes and File Colors .

Predefined scopes

PhpStorm provides a number of predefined scopes, for example:

- Project Files. This scope includes all the files within the project content roots (see Content Root and Configuring Content Roots). Libraries and SDKs are, generally, are not included in this scope.
- Problems. This scope includes the files within the project content roots in which syntactic errors are found.
- Project and Libraries. This scope includes all the files within the project content roots, and also all libraries and SDKs.
  In the Project tool window , this scope corresponds to the scope view All.
- Project Production Files. This scope is similar to the Project Files scope . The difference is that the test source roots are not included. In the Project tool window , this scope corresponds to the scope view Production.
- Project Test Files. This scope is limited to the project test source roots. In the Project tool window , this scope corresponds to the scope view Tests.
- Non-Project Files. This scope is available only as a view in the Project tool window . It is limited to libraries and SDKs.
- Changed Files. This scope corresponds to all changed files, that is, ones associated with all existing changelists .
- Default. This scope corresponds to the files associated with the changelist `Default` .
- Favorite ' `<name>` '. This scope corresponds to a list of favorite items with the specified name. See Managing Your Project Favorites .
- Open files. This scope corresponds to the files opened in PhpStorm editor.
- Current file. This scope corresponds to the file currently active in PhpStorm editor.
- Selected files. This scope corresponds to the files currently selected in PhpStorm (e.g. in the Project tool window).

Predefined scopes cannot be edited.

Library

A library is a collection of code used in your projects as is. This collection is presented in a set of files stored in an archive or directory (or in a number of archives and directories).

Libraries may optionally include API documentation which enables you to use inline documentation extracted from the source code and to view the API documentation right from PhpStorm.

Libraries are protected against any kind of modification, such as direct intentional editing or update result from refactoring. Libraries are also excluded from error checks. All this allows to save resource and improve performance and productivity.

JavaScript libraries can be configured at the global (IDE) or project level. The level of a library defines reusability of the library within the IDE.

In PHP development, you can use a third-party tool or framework as a library in several modes:

- Store the tool under the project content root.
- Store the tool under the PHP home directory and specify relative paths to it .
- Store the tool elsewhere and configure it as a custom command line tool .

In either case, you can use exploded and archived tool sources. PhpStorm recognizes most popular `ZIP` and `TAR.GZ` formats as well as the PHP native PHAR format.

All the `PHAR` files from the current project and the specified include path are shown in the project tool window under the Project View/Libraries/PHAR node and available for browsing right there. `PHAR` files are involved in indexing and therefore are used in autocompletion, search, reference resolution, etc.

Path Variables

To avoid complications when moving projects from one computer to another, PhpStorm provides path variables. (A path variable, obviously, is a variable whose value is an absolute path to a directory or file.)

Path variables are particularly useful when working with third-party libraries stored outside the project directory.

To illustrate, let's assume there is a project shared among a team of developers (e.g., through version control), and there is a library in this project defined at the project level.

Everything goes fine if the library is located under the project content root. This, however, is rarely the case.

A more typical situation is when the library location is external to the projects. Such a library is referenced by its absolute path and there's no guarantee that this path is the same on every one of the computers used by the team.

The obvious solution is to define a path variable for the library location. In such a case, the library path may be set individually on each of the computers.

To summarize, you should define path variables under the following set of conditions:

- There are third-party libraries in your project defined at the project level.
- These libraries are stored outside the project content root.
- There is a need to work with the project on more than one computer (e.g., share the project among the development team members through version control).

Ignored variables

If when opening a project, PhpStorm detects unresolved path variables, it asks you to define proper values for them. If for some reason you don't want to do that (e.g., you are not going to use the corresponding library or libraries), you have an option of adding the corresponding variables to a list of ignored variables.

There may also be other cases when the list of ignored variables is useful.

At the internal level, path variables are represented by strings in which the name of a variable is enclosed between a pair of dollar sign characters, for example, `$MY_PATH_VARIABLE$`.

Such a pattern, in principle, may occur in your project without the meaning of a path variable.

To tell PhpStorm that a string that starts and ends with a dollar sign character (e.g., `$SOME_STRING$`), actually, is not a path variable, you should add such a string (e.g., `SOME_STRING`) to the list of ignored variables.

Path variables and the list of ignored variables are configured on the Path Variables page of the Settings dialog.

Supported Languages

In this section:

- Supported languages
- Coding assistance

Supported languages

Development of a modern application involves using multiple languages, that is why PhpStorm is an IDE for polyglot programming. With the deep understanding of all the subtleties of the source code structure and syntax, PhpStorm extends its support to:

- JavaScript . Refer to the section JavaScript-Specific Guidelines .
- CoffeeScript . Refer to the section CoffeeScript Support .
- HTML / XHTML . Refer to the section Markup Languages and Style Sheets .
- XML . Refer to the section Markup Languages and Style Sheets
- XSLT .
- CSS : Less ; Sass ; Stylus , and Compass .

  Refer to Markup Languages and Style Sheets .
- PHP up to version 5.4.0 with support of syntax highlighting in `.ini` files. Refer to the section PHP-Specific Guidelines .
- SQL . Refer to the section Databases and SQL .

Coding assistance

Coding assistance in PhpStorm includes:

- Syntax and error highlighting. The color attributes are configurable in the Colors and Fonts | <language> pages of the Settings dialog.
- File templates for the supported languages that enable creating stub classes, scripts etc.
- Live templates for creating complicated code constructs.
- Code completion .
- Code generation .
- Code folding , formatting , and highlighting.
- Intention actions and quick fixes .
- Ability to view code hierarchy .
- Quick access to the API documentation .
- Using macros in the editor .
- Advanced search and replace facilities .
- Advanced means of navigation .
- Refactoring .
- Import Assistance
- The embedded local terminal where you can execute commands without leaving PhpStorm.

Besides editing assistance, PhpStorm features a JavaScript debugger .

Warning! Debugging for JavaScript applications is supported only in the Chrome browser.

For PHP projects, debugging and profiling performance with Xdebug or Zend Debugger is supported.

PHP Support

PHP support includes:

- Possibility to create PHP files and classes from templates .
- Full PHP 5.5 syntax support including generators, coroutines, the finally keyword, list in foreach, using empty() on the result of function calls and other expressions, class name resolution as scalar via the class keyword, and constant array/string dereferencing.
- Syntax highlighting .
- Error highlighting .
- Basic on-the-fly code completion .
- Resolution of `include` statements and file references, including references to `PHAR` archives.
  All the `PHAR` files from the current project and the specified include path are shown in the project tool window under the Project View/Libraries/PHAR node and available for browsing right there.
- Class Completion .
- Intention actions and quick fixes .
- Surrounding with code constructs `Ctrl+Alt+J` and `Ctrl+Alt+T`.
- Code inspections .
- Jump to declaration (`Ctrl+B`).
- Refactoring :
  - Rename (`Shift+F6`).
  - Move (`F6`).
  - Copy (`F5`).

PHP development support in PhpStorm is provided through the PHP plugin. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Syntax Highlighting

- Basic syntax highlighting:

```
class Concatenation {
    public function concatenate($string1, $string2) {
        $result = $string1.$string2;
        return $result;
    }
}
```

- Improved PHP syntax coloring for various constructs. There are separate colors for: function and method declarations; function calls, static and instance method calls; classes and interfaces, static and instance class fields:

```
<?php
class PersonRepository implements IRepository
{
    /** @var Db */
    protected $_db = null;

    function __construct($_db)
    {
        $this->_db = $_db;
    }

    public function getById($id)
    {
        return $this->_db->select('person', 'id = ?', array($id));
    }
```

Error highlighting

```
class Concatenation {
    public function concatenate($string1, $string2) {
        $result = $string1.$string2;
        return $result
    }                    Expected: semicolon
}
```

Basic on-the-fly code completion

```
class Concatenation {
    public function concatenate($string1, $string2) {
        $result = $string1.$string2;
        $
    }   $result string
    }   $string1
}       $string2
        $w Concatenation                    hierarchy.php
$w = I  $name string                        calligra.php
$w->c   $_ADODB_ACTIVE_DBS array            adodb-active-record.inc.php
```

Class Completion

```
class CompleteClass extends |
                            CompleteClass          hierarchy.php
                            Concatenation          hierarchy.php
                            UnitTest2              phpunit_test2.php
                            __dummyConverter       Converter.inc
```

Intention actions and quick fixes

```php
class Concatenation {
    public function concatenate($string1, $string2) {
        $result = $string1.$string2;
        return $rresult;
    }
}
```

- Resolve using PHPDOC annotation
- Typo: Accept 'rresult' as correct
- Typo: Rename to...

JavaScript and AJAX

PhpStorm enables creating rich Internet applications and Web applications by providing elaborate support of JavaScript and tight integration with AJAX and other adjacent frameworks and technologies.

Supported languages:

- JavaScript . Refer to the section JavaScript-Specific Guidelines .
- CoffeeScript . Refer to the section CoffeeScript Support .

Running and Debugging

In PhpStorm, you can run and debug various types of applications without leaving the IDE.

Each script, console, server, or test that you wish to run or debug from within PhpStorm, needs a special profile that specifies script name, working directory, and other important data required for running or debugging in different modes. PhpStorm comes with a number of such pre-defined profiles, or run/debug configurations .

Using the PhpStorm's debugger, you can find out the origin of the run-time errors and exceptions. The debugger enables you to execute your application step by step, examine program information related to variables, watches, or threads, and change your program without leaving the IDE.

Prior to launching the debugger session, you have to set breakpoints that cause the debugger to suspend application (or take some other actions), when such breakpoint is reached.

Generated output and console information during the running or debugging sessions are displayed in the Run tool window and Debug tool window respectively.

In this part you can find basic information about:

- Run/Debug Configuration
- Breakpoints

Refer to the sections Running , Debugging and Testing for the detailed descriptions of procedures.

Run/Debug Configuration

On this page:

- Basics
- Temporary configuration
- Permanent configuration
- Default run/debug configuration settings

Basics

PhpStorm enables using numerous run/debug configurations. Each run/debug configuration represents a named set of run/debug startup properties. When you perform run, debug, or test operations with PhpStorm, you always start a process based on one of the existing configurations using its parameters.

PhpStorm comes with a number of run/debug configuration types for the various running, debugging and testing issues. The user can create his/her own run/debug configurations of specific types.

Each run/debug configuration type has its own default settings. Whenever a new run/debug configuration of the respective type is created, it is based on these default settings.

Temporary configuration

A temporary run/debug configuration is automatically created every time you choose Run <item_name> or Debug <item_name> for an item without a permanent configuration. Temporary configurations can be saved as permanent.

Temporary configurations are marked with semi-transparent icons and are managed same way as the permanent configurations.

By default, 5 temporary configurations are allowed per project. You can change this limit via the Edit Configurations dialog.

Permanent configuration

A configuration of this type is explicitly created for a particular class or method. If there is no permanent configuration for an item, PhpStorm automatically creates a temporary configuration for it, when you choose Run <item_name> or Debug <item_name> on the context menu of this class or method.

Default run/debug configuration settings

The default run/debug configuration settings appear in the Run/Debug configurations dialog box under the node Defaults. They denote the settings that are used when a new run/debug configurations is created.

You can set the default settings for a specific configuration type that will become applicable to any run/debug configuration of this type created later. Changing defaults does not affect the existing run/debug configurations.

The process of editing the per-type default configuration settings is described here . The process of creating or editing the custom run-debug configurations is described on the page Creating or Editing Run/Debug configurations .

Breakpoints

Breakpoints are source code markers used to trigger actions during a debugging session.

In this part:

- Types of Breakpoints
- Breakpoints Icons and Statuses

Types of Breakpoints

On this page:

- Introduction
- Line breakpoint
- Temporary Line breakpoint
- Exception breakpoint

Introduction

PhpStorm lets you create breakpoints of several types. Each breakpoint type supported by PhpStorm addresses different debugging needs and has its own individual settings. Depending on the context, PhpStorm distinguishes between the following types of breakpoints:

- PHP line breakpoints can be set in the PHP context inside `*.php`, `*.html`, and files of other types.
- PHP exception breakpoints
- JavaScript line breakpoints can be set in the JavaScript context inside `*.js` or `*.html` files.
- JavaScript exception breakpoints

Line breakpoint

These breakpoints are assigned to lines of source code and are used to target a particular section for debugging.

Line breakpoints are triggered when the program reaches the specified line of source code, before it is executed. The line of code that contains a set breakpoint, is marked with a red stripe; once such line of code is reached, the marking stripe changes to blue.

If a file with breakpoints has been modified externally, for example, updated from a version control repository, or changed in an external editor, so that line numbers are changed, then the breakpoints will be moved accordingly. Note that PhpStorm should be running at the moment of such modification; otherwise, such changes will pass unnoticed.

Temporary Line breakpoint

These breakpoints are assigned to lines of source code and are used to target a particular section for debugging. When hit, such breakpoints are immediately removed.

Exception breakpoint

Exception breakpoints are triggered when the specified exception is thrown. Unlike the line breakpoints, which require specific source references, exception breakpoints apply globally to the exception condition, rather than to a particular code reference.

With PHP Exception Breakpoints, you can initiate the debugger at the start of the script and break on your own breakpoints or whenever an error or Exception of a given type occurs. PHP Exception breakpoints do not require configuring Xdebug for working in the Just-In-Time mode by setting `xdebug.remote_mode` to `jit`, see Debugging in the Just-in-Time Mode for details.

Breakpoints Icons and Statuses

On this page:

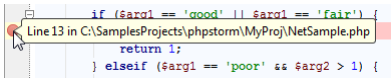
– Basics

– Breakpoint states and icons

Basics

When a breakpoint is set, the editor displays a breakpoint icon in the gutter area to the left of the affected source code. A breakpoint icon denotes status of a breakpoint, and provides useful information about its type, location, and action.

The icons serve as convenient shortcuts for managing breakpoints. Clicking an icon removes the breakpoint. Successive use of `Alt` - click on an icon toggles its state between enabled and disabled. The settings of a breakpoint are shown in a tooltip when a mouse pointer hovers over a breakpoint icon in the gutter area of the editor.



Breakpoint states and icons

The table below summarizes the possible breakpoint states:

Status	Line	Temporary Line	Exception	Description
Enabled				Shown at design-time or during the debugging session.
Disabled				Indicates that nothing happens when the breakpoint is hit.
Conditionally disabled				This state is assigned to breakpoints when they depend on another breakpoint to be activated.

When the button is pressed in the toolbar of the Debug tool window, all the breakpoints in a project are muted, and their icons become grey: .

Code Analysis

PhpStorm performs code analysis by applying inspections to your code. Numerous code inspections exist for PHP and for the other supported languages.

The inspections detect not only compiling errors, but also different code inefficiencies. Whenever you have some unreachable code, unused code, non-localized string, unresolved method, memory leaks or even spelling problems – you'll find it very quickly.

PhpStorm's code analysis is flexibly configurable. You can enable/disable each code inspection and change its severity , create profiles with custom sets of inspections, apply inspections differently in different scopes , suppress inspections in specific pieces of code, and more.

The analysis can be performed in several ways:

- By default, PhpStorm analyses all open files and highlights all detected code issues right in the editor. On the right side of the editor you can see the analysis status of the whole file (the icon in the top-right corner).
  When an error is detected, this icon is ; in case of a warning, it is ; if everything is correct, the icon is .
- Alternatively, you can run code analysis in a bulk mode for the specified scope, which can be as large as the whole project.
- If necessary, you can apply a single code inspection in a specific scope.

For the majority of the detected code issues, PhpStorm provides quick fix suggestions . You can quickly review errors in a file by navigating from one highlighted line to another by pressing `F2` `Shift+F2` .

PhpStorm analyses your projects on the various levels:

- "On-the-fly" analysis helps you fix problems as they arise as you type, using intention actions .
- Code inspections are intended to point out issues related to the program design.
- Highlighting level allows you to control the scope of problems highlighted in the current file.

Encoding

Different types of files use different ways to define encoding. PhpStorm recognizes encoding of files based on their contents.

Encoding has influence on the way PhpStorm reads or writes files. If a file has been modified but not yet saved, any changes in encoding affect file writing; if a file has not been modified, then reading is affected. PhpStorm suggests specific ways to change encoding of a file according to its type, using File Encodings Settings page, the Status bar, or the editor .

Note Encoding applies to directories and individual files. The encoding information saved in a file overrides the default encoding; encoding of a file or subdirectory overrides encoding settings on the higher levels.

EncodingCan be changed in	
File encoding is specified within the file, for example, in XML.	If a file contains explicit encoding declaration, you can change it in the Editor . In this case PhpStorm provides code completion.
File encoding is defined by BOM.	In this case, you can't change encoding with which PhpStorm reads the file, but it is still possible to change encoding for writing such file.
UTF characters are detected in the file contents.	PhpStorm provides an option that automatically changes file encoding to UTF , if the file contents can be reasonably interpreted as UTF. This option only works for reading; a file can be saved with any encoding.
Encoding cannot be found out from the file content.	In this case, the default encoding is the one defined by the IDE encoding in the File Encodings page of the Settings dialog. You can change it for multiple files and directories , or for a single file .

Dialogs

This part contains descriptions of the following dialogs:

- Add Composer Dependency Dialog
- Analyze Stacktrace Dialog
- Bookmarks Dialog
- Breakpoints
- Choose Local Paths to Upload Dialog
- Color Picker
- Code Duplication Analysis Settings
- Composer Settings Dialog
- Confirm Drop dialog
- Create New PHPUnit Test
- New Project Dialog
- Create Table and Modify Table dialogs
- CSV Formats Dialog
- Differences viewers
- Docker Registry Dialog
- Edit as Table: <file_Name> Format Dialog
- Edit Macros Dialog
- Edit Project Path Mappings Dialog
- Evaluate Expression
- Export Threads
- Export to HTML
- File Cache Conflict
- Find and Replace in Path
- Find Usages Dialogs
- Generate Instance Document from Schema Dialog
- Generate Schema from Instance Document Dialog
- Incoming Connection Dialog
- Map External Resource Dialog
- Non-Project Files Protection Dialog
- New Project from Existing Files Wizard
- Open Task Dialog
- Optimize Imports Dialog
- Override Server Path Mappings Dialog
- PhpStorm License Activation Dialog
- Manage Project Templates Dialog
- Print
- Productivity Guide
- PSI Viewer
- Pull Image Dialog
- Push Image Dialog
- Recent Changes Dialog
- Refactoring Dialogs
- Reformat File Dialog
- Reformat Code on Directory Dialog
- Register New File Type Association Dialog
- Run/Debug Configurations
- Save Project as Template Dialog
- Select Path Dialog
- Specify Code Cleanup Scope Dialog
- Specify Code Duplication Analysis Scope
- Specify Inspection Scope Dialog
- Structural Search and Replace Dialogs
- Web Server Debug Validation Dialog

Add Composer Dependency Dialog

Tools | Composer | Add Dependency

In this dialog box, appoint the packages to be added to the current project.

Item	Description
Available packages	From this list, select the package that you need in your project. The list shows all the available packages, however, the packages that have been already installed are marked with a tick. Use the search field, if necessary: start typing the search string, as you type, the list dynamically reduces to show the packages that match the entered pattern. The Description read-only text box briefly explains the functionality of the selected package.
Version to install	From this drop-down list, select the package version. The contents of the list depend on the specific package.
Install	When you click this button, PhpStorm starts downloading the appointed package. When the process is completed, PhpStorm creates a new folder under the Vendor node stores the downloaded package in it, and adds the package to the list in the `require` section of `Composer.json`.
Settings	In this area, specify the advanced installation options: – Path to PHP executable: in this field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens. – Command line parameters: in this text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have a dependency added to the `require-dev` section instead of the default `require` section type `-dev`. For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md.

Analyze Stacktrace Dialog

Tools | Analyze Stacktrace

Use this dialog box to reach a navigable console stack trace for external applications. From each message in this stack trace, you can navigate right to the source code that cased the reported problem.

Item	Description
Put a stack trace or a complete thread dump here	Paste here the external stack trace or thread dump.

Bookmarks Dialog

Navigate | Bookmarks | Show Bookmarks

`Shift+F11`

Use this dialog to navigate between bookmarks, and manage the collection of anonymous bookmarks and bookmarks with mnemonics in a project.

Toolbar options

Item	Shortcut	Description
	`Ctrl+Enter`	Click to add/edit description for the selected bookmark.
	`Alt+Delete`	Click to delete the selected bookmark.
	`Alt+Down` `Alt+Up`	Use these buttons to reorder bookmarks.

The left pane of the Bookmarks dialog displays the list of bookmarks created with the brief description, if any. The order of bookmarks in the collection defines the order in which Navigate | Bookmarks | Next/Previous Bookmark command visits the bookmarks. The right pane displays the preview of the file where the selected bookmark is toggled.

To close the dialog box, press `Escape`.

Breakpoints

Run | View Breakpoints

`Ctrl+Shift+F8`

In this section:

- Toolbar
- Breakpoint options
- Context menu commands
- Speed search of a breakpoint

Toolbar

Item	Tooltip and shortcut	Description
+	Add Breakpoint `Alt+Insert`	Click to show the list of available breakpoint types . Select the desired type to create a new breakpoint.
−	Remove Breakpoint	Click this button to remove selected breakpoints.
	Group by File	Click this button to display breakpoints under their respective files, rather than under their types:

Breakpoint options

The controls of this part of the Breakpoints dialog depend on the type of the selected breakpoint.

Option	Description	Types of breakpoints
Suspend	Select this check box to suspend program when a breakpoint is hit. If the check box is not selected, the program is not suspended.	All types
Condition	Select this check box and specify a condition for hitting a breakpoint in the text field. A condition is a Boolean expression. This expression should be valid at the line where the breakpoint is set, and is evaluated every time the breakpoint is reached. If the evaluation result is `true` , user-selected actions are performed. If the result is `false` , the breakpoint does not produce any effect. If the Debugger cannot evaluate the expression, it displays the Condition evaluation error message. You can select whether you would like to stop at this breakpoint or ignore it. To the right of the Condition field, there is the button (`Shift+Enter`) that opens the multiline editor.	All types
Log message to console	Select this check box if you want a log message to be displayed in the console output when the breakpoint is hit.	All types
Evaluate and log	Select this check box if you wish to evaluate a certain expression at this breakpoint and to export result to the console output. To the right of this field, there is the button (`Shift+Enter`) that opens the multiline editor. Tip If the expression to be evaluated is incorrect when a particular breakpoint is reached, the console output displays an error message: `Unable to evaluate expression <your_expression>`	Line breakpoints
Remove once hit	Select this check box, if the you want the breakpoint to be deleted after hitting it.	All types
Disabled until selected breakpoint is hit	From the drop-down list, select the breakpoint in question. The option None corresponds to the always enabled breakpoint. Besides that, you can also choose the behavior of this breakpoint, when the selected one is hit: − Disable again − Leave enabled	All types

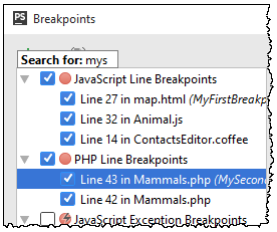
Context menu commands

Command	Description
Move to group	Point to this command to move the selected breakpoint to a new group , to one of the existing groups(`<group name>`), or out of a group (`<no group>`).
Edit description	Choose this command to enter or change description of a breakpoint .

Speed search of a breakpoint

To find a particular breakpoint

− Start typing address or description of the target breakpoint:



PhpStorm highlights the line with the matching address or description.

Choose Local Paths to Upload Dialog

The dialog box opens when you click the Upload to Web Server check box in the Run/Debug Configuration dialog box and click the Browse button ... next to it.

Use the dialog box to configure a list of folders and files to be uploaded according to previously defined mappings and to edit these mappings, if necessary.

Item	Description
Add	Click this button to have a new row added to the list and specify the item to upload. Type the path manually or click the Browse button ... to select the desired location in the Choose Path dialog box, that opens.
Remove	Click this button to remove the selected entry from the list.
Edit Mappings	Click this button to open the Deployment dialog box, where you can define mappings between local folders and the corresponding paths on the FTP/SFTP server.

Color Picker

Settings | Editor | Colors and Fonts

`Ctrl+Shift+A`

Use this dialog to find the exact RGB, HSB and hex values of any color component.

Item	Description
	Click this eyedropper to navigate to the object and obtain its color RGB or HSB as well as its color hex values.
Color mixer	Use this area to select a color.
Stack of colors	This area displays colors that were last selected.
Brightness	Move the slider upwards to make to colors in the Color mixer area brighter; move the slider down to make colors darker.
Choose	Click this button to select a color from the Color Mixer area, or the Stack of colors.
Cancel	Click this button to omit saving the color information.

Code Duplication Analysis Settings

Code | Locate Duplicates - OK

Use this dialog to define the sensitivity of search, and set limitation that will help you avoid reporting about every similar code construct. Your preferences are specified in a language-specific context.

ItemDescription	
CSS	– Do not show duplicates containing less than <number> CSS properties : Set the size of duplicated language constructs that are shown in the results window.

CoffeeScript	– Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	– Anonymize Functions
	– Anonymize Literals
	
	– Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
	– Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
	The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

ECMA Script level 4	– Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	– Anonymize Functions
	– Anonymize Literals
	
	– Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
	– Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
	The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

HTML	– Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).

JavaScript	– Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	– Anonymize Functions
	– Anonymize Literals
	
	– Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
	– Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
	The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

JSON	– Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`
	
	`var test01 = function(a,b){`
	`return (a*b)`
	`}`

- Anonymize Functions
- Anonymize Literals

- Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
- Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
  The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

PHP

- Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:

```
var test01 = function(a,b){
return (a*b)
}

var test01 = function(a,b){
return (a*b)
}
```

- Anonymize Functions
- Anonymize Literals

- Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
- Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
  The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

TypeScript

- Anonymize Variables: when this check box is selected, two identical functions that use different variable names are considered duplicates, for example:

```
var test01 = function(a,b){
return (a*b)
}

var test01 = function(a,b){
return (a*b)
}
```

- Anonymize Functions
- Anonymize Literals

- Do not show duplicates simpler than: Set the size of duplicated language constructs that are shown in the result window. By default, the constructs less than 10 units are not included (and this limitation cannot be changed).
- Anonymize uncommon subexpressions simpler than: Set the value of the subelements within language constructs that can be considered similar, to show the construct as duplicate in the result window. The larger is the number, the larger constructs are taken as similar by PhpStorm.
  The values are set as arbitrary weights based on the element size calculated with additive algorithm. The larger is the element, the higher is the calculated value.

XHTML

- Do not show duplicates containing less than <number> tags : Set the size of duplicated language constructs that are shown in the results window.
- Anonymize values of tags and attributes

XML

- Do not show duplicates containing less than <number> tags : Set the size of duplicated language constructs that are shown in the results window.
- Anonymize values of tags and attributes

Composer Settings Dialog

Tools | Composer | Init Composer

In this dialog box, choose the instance of Composer to use in the current project among all the instances configured in PhpStorm. For more details, see Enabling and configuring the use of Composer Dependency Manager in PhpStorm .

Item	Description
Path to composer.phar	In this text box, specify the location of the `composer.phar` archive to use in the current project. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens. PhpStorm parses the contents of the specified file for Composer commands.
Click here to download from getcomposer.org	Click this link to have PhpStorm download `composer.phar` from the official storage and specify the folder to store the archive in. This instance of Composer will be available in the current project only. To use it in the command line mode, configure it as a command line tool .

Confirm Drop dialog

From the Database tool window :

Edit | Drop, Drop from the context menu, or `Delete`.

Click OK to remove the selected item or items, or click to copy the SQL statement or statements into the database console .

Item	Description
SQL Preview	The statement or statements to be run to remove the selected item or items. If necessary, you can edit the statements right in this pane. The statements are executed when you click OK.
	Copy the statements into the corresponding database console and close the dialog.

Create New PHPUnit Test

File | New | PHPUnit Test

Project tool window | Context menu of a class | New | PHPUnit Test

Use this dialog to configure generation of PHPUnit test class stubs.

The dialog box is available when the PHPUnit tool is installed on your machine and enabled in PhpStorm. For more information, see http://www.phpunit.de/manual/current/en/installation.html and Enabling PHPUnit Support .

Item	Description
Class To Test	In this area, specify the production class to generate a test class for. Type the fully qualified name of the class in the Fully Qualified Name text box. The specified name will be used to propose the Test Class Name. To use completion, press Control+Space and choose the relevant production class from the pop-up list.
Test Class	– Name: in this text box, specify the name of the test class to generate. PhpStorm automatically composes the name from the production class name as follows: `<production class>Test.php` . – Namespace: in this text box, specify the namespace the test class will belong to. PhpStorm completes the namespace automatically based on the value of the Directory text box. When the Directory text box is changed, the value of the Namespace text box is changed accordingly. To use completion, press Control+Space and choose the relevant namespace from the pop-up list. – Directory: in this text box, specify the directory where the file with the test class will be stored. By default, it is the directory where the production class is stored. To specify another folder, click the Browse button and choose the relevant folder in the dialog box that opens. To use completion, press Control+Space and choose the relevant folder from the pop-up list. – File Name: in this text box, specify the name of the file that will be created for the test class definition. By default, the name is the same as the test class name. However, if several production classes are defined in one single file, for each generated test class PhpStorm will create a separate file.

New Project Dialog

File | New Project

Welcome Screen | Create New Project

Use this dialog box to create empty projects or generate project stubs for developing PHP and Web applications.

The left-hand pane of the dialog box shows a list of project types for which stubs can be generated. The contents of the right-hand pane depend on the selected project type.

The project types are grouped under the PHP-Specific and Web categories. If you have any project templates saved, they are listed under the User Defined category.

PHP empty project

Choose this option to get just a project folder without any contents. See New Project: PHP Empty Project for a description of the fields in the right-hand pane.

PHP-specific project types

- Composer Project: choose this option to have a project stub created using the Composer template . See New Project: Composer Project for a description of the fields in the right-hand pane.
- Drupal Module: choose this option to have a Drupal module created and set up. See New Project: Drupal Module for a description of the fields in the right-hand pane. The option is available only if the Drupal Support plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
- WordPress Plugin: choose this option to get a project stub for developing applications using WordPress, see Using WordPress Content Management System for details. In the right-hand pane, type the project location and specify the folder where WordPress is installed. This folder should contain the `wp-admin` and `wp-includes` subdirectories.

  The option is available only if the WordPress Support plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
- App Engine Project: choose this option to get a project stub for developing applications that will run in the Google PHP Runtime Environment , see Using Google App Engine for PHP for details. See for a description of the fields in the right-hand pane.
  The option is available only if the Google App Engine for PHP plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Web

- HTML5 Boilerplate: choose this option to have the project structure set up and some sources generated based on the HTML5 Boilerplate template. See New Project: HTML5 Boilerplate for a description of the fields in the right-hand pane.
- Web Starter Kit: choose this option to have the project structure set up and some sources generated in accordance with the Web Starter Kit requirements. See New Project: Web Starter Kit for a description of the fields in the right-hand pane.
- React Starter Kit: choose this option to have the project structure set up and some sources generated according to the React requirements. See New Project: React Starter Kit for a description of the fields in the right-hand pane.
- Twitter Bootstrap: choose this option to have the project structure set up and some sources generated based on the Twitter Bootstrap template. See New Project: Twitter Bootstrap for a description of the fields in the right-hand pane.
- Foundation: choose this option to have the project structure set up and some sources generated based on the Foundation framework template. See New Project: Foundation for a description of the fields in the right-hand pane.
- AngularJS: choose this option to have the project structure set up and some sources generated based on the AngularJS framework template.
- Dart Web Application: choose this option to have the project structure set up and some sources generated for a Dart application.
- Node.js Express App: choose this option to have the project structure set up and some project sources generated based on the Express framework. See New Project: Node.js Express App for a description of the fields in the right-hand pane.

  The option is available only if the Node.js plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
- PhoneGap/Cordova App: choose this option to have the project structure set up and some sources generated based on the PhoneGap , Apache Cordova , and Ionic frameworks. See New Project: PhoneGap/Cordova for a description of the fields in the right-hand pane.
- Meteor App: choose this option to have the project structure set up and some sources generated based on the Meteor frameworks. See New Project: Meteor Application for a description of the fields in the right-hand pane.

User Defined

This list is displayed only if you have previously saved an project templates. Choose the relevant item based on its description, see Saving Projects as Templates for details.

New Project: PHP Empty Project

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose PHP Empty Project from the list in the left-hand pane of the Create New Project dialog .

Item	Description
PHP Language Level	In this drop-down list, specify the PHP functionality scope to get coding assistance for. Each functionality scope is associated with PHP version that supports this functionality. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported. No correlation between the PHP version used in the project and the language level is enforced. Although the language version of each interpreter is detected automatically, you can still tell PhpStorm to provide you with coding assistance that corresponds to another language level. However, if you attempt to use a code construct that is not supported by the specified language level, PhpStorm suggests a Switch to PHP <version> quick-fix .
Interpreter	From this drop-down list, choose the PHP interpreter to use in the current project. The list contains all the currently configured local and remote PHP interpreters. See Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters for details.
Include path	This area shows a list of paths to PHP-related items below the PHP home directory. The specified include paths will be used: – By the `require()` , `include()` , `fopen()` , `file()` , `readfile()` , and `file_get_contents()` functions when looking for files to use. – By PhpStorm when resolving references to included files. Use the Add + and Remove — buttons to manage the contents of the list. Use the Up and Down buttons to change the order of items in the list.

New Project: Composer Project

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Composer Project from the list in the left-hand pane of the Create New Project dialog .

Use the dialog box to have PhpStorm create a project with Composer-specific structure. To do this, appoint the Composer instance to use and install the package you need in the project.

Item	Description
Composer.phar	In this area, appoint the `composer.phar` command definition file to use in project creation. – Use existing composer.phar: choose this option to use commands from a previously downloaded `composer.phar` and specify it location in the text box. – Download composer.phar from getcomposer.org: choose this option to have a new instance of Composer downloaded. The `composer.phar` file will be saved under the project root folder specified in the Location text box of the New Project Dialog dialog.
Package	In this area, specify the package to install during the project creation. – Available packages: from this list box, select the package to install. Use the search field, if necessary: start typing the search string, as you type, the list dynamically reduces to show the packages that match the entered pattern. – Description: this read-only text box briefly explains the functionality of the selected package. – Version to install: from this drop-down list, select the package version. The contents of the list depend on the specific package.
Settings	In this area, specified advanced settings for generating a project stub and installing packages (dependencies): – Path to PHP executable: in this field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens. – Command line parameters: in this text box, type the additional command line parameters to adjust the add dependency procedure. For example, to have a package added to the `require-dev` section instead of the default `require` section type `-dev` . For more information about Composer command line options during installation, see https://getcomposer.org/doc/03-cli.md .

New Project: Drupal Module

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Drupal Module from the list in the left-hand pane of the Create New Project dialog .

Use this dialog box to generate and set up a module stub in compliance with the Drupal requirements.

Item	Description
installation path	In this text box, specify the root folder of the installation. Type the path manually or click the Browse button ... and select the relevant folder in the dialog box that opens.
Set up PHP | Include paths	– Select this check box to have Drupal include paths automatically configured for the project. The following paths will be added to the Include Paths list on the PHP page of the Settings dialog box: 1. `<drupal installation root>/includes` 2. `<drupal installation root>/modules` 3. `<drupal installation root>/sites/all/modules` The include paths are added to the list after you leave the dialog box. – Clear the check box to configure the include paths manually.
Version	From this drop-down list, choose the version of Drupal to use, the supported versions are 6, 7, and 8 .

New Project: Google App Engine for PHP

File | New Project - PHP - App Engine Project

Use this dialog box to configure Google App Engine for PHP project settings.

Item	Description
Application ID	In this text box, type the identifier of your application as you specified it on the Create application page.
SDK directory	In this text box, specify the path to the folder where the Google App Engine SDK for PHP is installed.
Python executable	In this text box, specify the location of the Python executable file. Type the path manually or click the Browse button ... and choose the executable file in the dialog box that opens.

New Project: HTML5 Boilerplate

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose HTML5 Boilerplate from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.

Create New Project: Joomla! Integration Project

PhpStorm

New Project: Web Starter Kit

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Web Starter Kit from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.

New Project: React Starter Kit

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose React Starter Kit from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.

New Project: Twitter Bootstrap

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Twitter Bootstrap from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.

New Project: Foundation

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Foundation from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Location	In this text box, specify the path to the project folder where the project-related files will be stored.
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.

New Project: Node.js Express App

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Node.js Express App from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Node interpreter	In this field, specify the location of the Node.js executable file. In most cases, PhpStorm detects the Node.js executable and fills in the field automatically.
Npm executable	In this field, specify the location of the Node.js package manager file `npm.cmd` . In most cases, PhpStorm detects the Node.js executable and fills in the field automatically.
Version	From this drop-down list, choose the version of the template in accordance to which the stub will be generated. Click to refresh the list of available template versions.
Template	From this drop-down list, choose the template engine to use. The available options are: – Jade - haml.js successor. – EJS - embedded JavaScript. – Hogan.js . – Handlebars .
CSS	From this drop-down list, choose the CSS to use preprocessor to use. The available options are: – Plain CSS – Stylus – Less – Compass . – Sass .

New Project: Meteor Application

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Meteor App from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Meteor	In this text box, specify the location of the Meteor executable file (see Installing Meteor).
Template	From this drop-down list, choose the sample to generate. To have a basic project structure generated, choose the Default option.
Filename	In this text box, type the name for the mutually related `.js` , `.html` , and `.css` files that will be generated. The text box is available only if the Default sample type is selected from the Template drop-dow list.

New Project: PhoneGap/Cordova

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose PhoneGap/Cordova App from the list in the left-hand pane of the Create New Project dialog .

Item	Description
PhoneGap/Cordova	In this text box, specify the location of the executable file `phonegap.cmd` , or `cordova.cmd` , or `ionic.cmd` (see Installing PhoneGap/Cordova/Ionic).

New Project: Yeoman

File | New Project

Welcome Screen | Create New Project

The dialog box opens when you choose Yeoman from the list in the left-hand pane of the Create New Project dialog .

Item	Description
Location	In this text box, specify the path to the project folder where the project-related files will be stored.
Generators	In this area, create a list of Yeoman generators you need and select the one to use during the current project generation. − To add a generator to the list: 1. Click Install Generator. 2. From the dialog box that opens showing all the available generator packages, select the required package in the left-hand pane and click the Install Generator button that appears in the right-hand pane. You can install several packages one after another without leaving the dialog box. When the installation is over, click Close to return to the list of generators which is already expanded with the newly added package. − To create a project using a generator: 1. In the Location field, specify the folder where the new project will be created. Type the path manually or click the ... and select the folder in the dialog box that opens. 2. If necessary, in the Options text box, specify additional flags to customize the project creation. The set of available flags depends on the selected generator. 3. Select the required generator from the list and click Next. 4. Specify the required settings in the New Project wizard that starts. The number of pages and their contents depend on the chosen generator. 5. On the last page of the wizard, select or clear the Run npm install&bower install check box to specify whether you want to run Node Package Manager and Bower to install the packages that are required for developing the new project. 6. Click Next and choose to open the new project in the current window or in the new one.
Options	In this text box, specify additional flags to customize the project creation. The set of available flags depends on the selected generator.
Configure Node.js and Yeoman	Click this link to open the Yeoman dialog box and change the settings of Node.js and Yeoman.

Create Table and Modify Table dialogs

- Upper part: GUI
- SQL Script
- Options

Upper part: GUI

The upper part of the dialog provides a GUI for defining the table structure, and associated constraints and indexes.

SQL Script

The pane under SQL Script shows the statement or statements to be run to achieve the result you have specified using the GUI.

You can use this pane just as a preview. You can also edit the statement or statements right there.

Options

Item	Description
Execute in database	Change data structure in your database by running the statement or statements. Execute. Execute the statements right away. If associated changes in database consoles are possible, the corresponding preview is shown. See Previewing changes. Preview. Preview potential associated changes prior to executing the statements.
Replace existing DDL	Replace the definition (usually, a `CREATE` statement) in the corresponding database console or SQL file with the statements shown under SQL Script. Replace. Perform the replacement right away. If associated changes are possible, the corresponding preview is shown. Preview. Preview potential associated changes prior to performing the replacement.
Open in editor	Copy the statements into the corresponding database console or SQL file. - Modify existing objects. In the generated set of statements, `ALTER` statements are preferred to `CREATE` statements. - Create modified objects. The changes you have made using the GUI are translated into a minimal set of `CREATE` and other statements. - Create all objects. The generated set always includes the `CREATE TABLE` statement, as if the whole table were created anew. To Editor. Switch to the corresponding editor tab right away. If associated changes are possible, the corresponding preview is shown prior to copying the statements. Preview. Preview potential associated changes.

CSV Formats dialog

To access this dialog:

- From any of the table views:

  Right-click the table and select Data Extractor | Configure CSV Formats.

- From the Database tool window :

  Right-click the table or view of interest, select Dump Data to File | Configure CSV Formats.

This dialog lets you specify the settings for converting table data into delimiter-separated values formats (e.g. CSV, TSV) and vice versa.

When working on the conversion settings, use the preview in the right-hand part of the dialog.

Item	Description
Formats	The list of the available delimiter-separated values formats is shown. Each format is a named set of corresponding conversion settings. Select the format whose settings you want to view or edit. Use +, −, ↑ and ↓ to create, delete and reorder the formats; to create a copy of the selected format.
Value separator	Select or type the character for separating individual values.
Row separator	Select or type the character for separating rows.
Null value text	The text to be used as a value if a cell contains `null` (an unknown value).
Add row prefix/suffix	Row prefix and suffix are character sequences which in addition to the row separator indicate the beginning and end of a row. If necessary, click the link and specify the row prefix and suffix in the fields that appear.
Quotation	Each line in the area under Quotation is a quotation pattern (see Quote values). A quotation pattern includes: − The left quotation character, the one inserted before a value. − The right quotation character, the one inserted after a value; usually, the same as the left quotation character. − An escape method or character for the cases when the quotation character is part of a value. E.g. Escape: duplicate means that if a quotation character occurs within a value, it is doubled. (You can specify your own escape character instead.) If there is more than one pattern, the first of the patterns is used. Use +, −, ↑ and ↓ to create, delete and reorder the patterns. To start editing an existing pattern, just click the pattern of interest.
Quote values	Specify in which cases the values should be quoted (i.e. enclosed within quotation characters). − When needed. A value is quoted only if it contains the value and/or the row separator. − Always. Any value is quoted in its text representation.
Trim whitespaces	If this check box is not selected, the Unicode whitespace characters that precede and follow the value separators are treated as parts of the corresponding values. If this check box is selected, the corresponding whitespace characters are ignored or removed.
First row is header	If this check box is selected, the first row is treated as containing column names. The settings that appear under Header Format have the same meanings as the ones above but are applied to the first row.
First column is header	If this check box is selected, the first column is treated as containing row names.

Differences viewers

- Differences viewer for files
- Differences viewer for folders
- Differences viewer for table structures
- Differences viewer for tables

Differences viewer for files

Project tool window | context menu of a file | Compare File with Editor

Project tool window | context menu of a file | Compare Files

Version Control tool window | Repository and Incoming tabs |

Version Control tool window | context menu of a folder or file | Show Diff

In this section:

- Basics
- Diff & Merge viewer
- Keyboard shortcuts

Basics

This dialog is displayed every time you compare two files, or compare two versions of a file (local changes or changes between local files and their revisions in a remote repository). You can compare files of any types, including binaries and `.jar` files.

Note that you can open the differences viewer without running PhpStorm. To do this, execute the following command:

```
<path to PhpStorm
                    executable
                    file> diff <path_1> <path_2>
```

The differences viewer provides a powerful editor that enables code completion, live templates, etc.

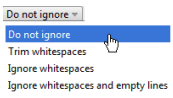
where `path_1` and `path_2` are paths to the files you want to compare.

Diff & Merge viewer

Item	Tooltip and Shortcut	Description
	Previous Difference / Next Difference `Shift+F7` `F7`	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to click the arrow buttons `F7` / `Shift+F7` once more and compare other files, depending on the Go to the next file after reaching last change option in the Differences Viewer settings . This behavior is supported only when the Differences Viewer is invoked from the Version Control tool window.

	Compare Previous/Next File
	
	`Alt+Left`
	`Alt+Right`
		Click these buttons to compare the local copy of the previous/next file with its update from the server.
		Tip These buttons are available only when the Differences Viewer is invoked from the Version Control tool window.

	Go To Changed File `Ctrl+N`	Click this button to display the path to the compared file. Tip This button is available only when the Differences Viewer is invoked from the Version Control tool window.
	Jump to Source `F4`	Click this button to open the selected file in the active pane in the editor. The caret will be placed in the same position as in the Differences Viewer.
Viewer type		Use this drop-down list to choose the desired viewer type. The side-by-side viewer has two panels; the unified viewer has one panel only. Both types of viewers enable you to – Edit, for example, type text. Note that one can change text only in the right-hand part of the default viewer, or, in case of the unified viewer, in the lower ("after") line. – Perform the Apply/Append/Revert actions.

Whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

Highlighting mode		Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted

	Highlight lines Do not highlight Highlight words Highlight lines Highlight split changes	– Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Collapse unchanged fragments	Click this button to collapse all unchanged fragments in both files. The amount of non-collapsible unchanged lines is configurable in the Diff & Merge settings page.
	Synchronize scrolling	Click this button to simultaneously scroll both differences panes; if this button is released, each of the panes can be scrolled independently.
	Editor settings	Click this button to reveal the drop-down list of available check commands. Select or clear these check commands to show or hide line numbers, indentation guides, white spaces, and soft wraps. Note that same commands are duplicated in the context menu of the differences viewer gutter. Right-click the gutter to show the menu of check commands.
	Include into commit Alt+I	This checkbox only appears if you invoke the Differences Viewer from the Commit Changes dialog with multiple changed files (all of which are deselected), and you explore the differences between them and hit the last difference in a file. Select this checkbox if you want to include the file you've reviewed into the commit.
	Move to Another Changelist F6	This button only appears if you invoke the Differences Viewer from the Commit Changes dialog with multiple changed files (all of which are deselected), and you explore the differences between them and hit the last difference in a file. Click this icon to move the file you've reviewed to another changelist.
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools settings page. This button only appears on the toolbar when the Use external diff tool option is enabled in the External Diff Tools settings page.
	Help F1	Click this button to show the corresponding help page.
	Ctrl+Tab	Use this keyboard shortcut to switch between the panes of the Differences viewer. The active pane has the cursor.
» «		Use these chevron buttons to apply differences between panes (in case of the side-by-side viewer) or between lines (in case of the unified viewer). The chevron buttons can change their behavior: – Click » « to apply changes. This behavior is the default one. – Press Ctrl to change » « to ⇙ or ⇘ and append changes.
Merge actions		
/ /	Compare Left and Middle/Middle and Right/Left and Right Contents	Click these buttons to compare left/middle/right parts in a new window.
	Apply All Non-Conflicting Changes	Click this button to apply all non-conflicting changes. You can also make this behavior automatic, by selecting the check box Automatically apply non-conflicting changes in the Diff & Merge page of the Settings/Preferences dialog.
/	Apply Non-Conflicting Changes from the Left/Right Side	Click these buttons to merge non-conflicting changes from the left/right parts of the dialog.

Keyboard shortcuts

Keyboard shortcut	Description
Ctrl+Shift+D	Use this keyboard shortcut to show the popup menu of the most commonly user diff commands.
Ctrl+Tab	Use this keyboard shortcut to switch between the left and the right panes.
Ctrl+Shift+Tab	Use this keyboard shortcut to select the position obtained by Ctrl+Tab in the opposite pane.
Ctrl+Z / Ctrl+Shift+Z	Use this keyboard shortcut to undo/redo a merge operation. Conflicts will be kept in sync with the text.

Differences viewer for folders

Project tool window | context menu of a folder | Compare Directory with

Project tool window | context menu of two selected folders | Compare Directories

Project tool window | context menu of a folder | Sync with Deployed to

Remote Host tool window | context menu of a folder | Sync with local

Tools | Deployment | Sync with local

Database tool window | context menu of two selected items | Compare

This window is displayed every time you explore differences between:

- Two local directories (local history for folders, recent changes, or version control).
- A remote folder and its local version .

In this dialog box, explore the detected differences and synchronize the compared items.

> **Tip** You can also open the difference viewer without running PhpStorm. This is done through the following command: `<path to PhpStorm executable file> diff <path_1> <path_2>` where `path_1` and `path_2` are paths to the folders in question.

In this section:

- Toolbar
- List of items
- Differences pane
- Diff viewer
- Context menu

Toolbar

Item	Tooltip and shortcut	Description	Available for
	F7 Shift+F7	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to press F7 / Shift+F7 once more and compare other files. **Tip** This feature becomes available only when the Differences Viewer is invoked from the Version Control tool window.	Version control
	Jump to Source F4	Click this button to open file in the editor's active tab. The caret will be placed in the same position as in the Differences Viewer.	All
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools page. This button only appears on the toolbar, when the check box Use external diff tool is selected in the External Diff Tools page.	All
	Help F1	Click this button to show reference page.	All
	Refresh F5	Click this button to refresh the contents of the differences viewer.	All
	Show new files on left side	Press this toggle button to have PhpStorm show the items, that are present in the first of the compared directories or database objects and are missing in the second one.	All
	Show difference	Press this toggle button to have PhpStorm show items that are present in both folders or database objects but differ in contents, or timestamp, or size.	All
	Show equals	Press this toggle button to have PhpStorm show the items that are present in both directories or objects and have the same contents, or timestamp, or size, depending on the parameter for comparison specified in the Compare by drop-down list.	All
	Show new files on right side	Press this toggle button to have PhpStorm show the items, that are present in the second of the compared directories or database objects and are missing in the first one.	All
Compare by		In this drop-down list, select the parameter to be used for comparison. The available options are: – Contents – Size – Time stamp	Local folders Local-remote folders
	Synchronize Selected	Click this button to have PhpStorm apply the specified action to the selected pair of items. Actions to be performed are shown in the * field.	All

	Enter Synchronize All	Click this button to have PhpStorm apply the specified action to all the pairs of items in the list. Actions to be performed are shown in the * field.	All
	Ctrl+Enter Hide excluded files	Click this button to suppress showing files excluded from synchronization .	Local-remote folders
Filter		Type the filtering string (for example, file or table name). Use * wildcard to replace any number of arbitrary characters. Note that filter applies on pressing Enter .	All
Path		These fields show the paths to the folders being compared. To change a directory, click the Browse button ... and specify another directory in the dialog that opens .	Local folders Local-remote folders
		These read-only fields show the names of the data sources or tables being compared.	Data sources

List of items

The list shows the items from the compared objects that meet the comparison criterion specified in the Compare by drop-down list and the filtering criteria specified through the toolbar buttons .

Item	Description	Available for
Name	These read-only fields show the names of files, data source tables, or table fields under the object specified in the Path or fields.	All
Size	These read-only fields show the sizes of files under the folders being compared.	Local folders Local-remote folders
Date	These read-only fields show the timestamps of files under the folders being compared.	Local folders Local-remote folders
*	The icon in this field indicates the action that will be applied to the pair of items in the current line upon clicking the Synchronize Selected or Synchronize All toolbar button. To change the currently selected action, click the icon. **Icon / Action** : Copy the item in the left side to the right side, possibly overwriting the contents of the corresponding target item, if it already exists. : Copy the item in the right side to the left side, possibly overwriting the contents of the corresponding target item, if it already exists. : The items are treated identical with regard to the selected criterion of comparison. No action will be performed by default. : The items differ with regard to the selected criterion of comparison. No action will be performed by default. Explore the differences in the Differences Pane and change the intended action by clicking the icon. : The item is present only in one of the folders and will be removed.	All

Differences pane

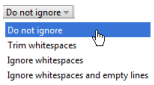
The differences pane is displayed only for files, data source tables, or table fields with the same names, which exist on both sides. For the files and DB objects that exist on one side only, the contents of the selected file/DB object is displayed.

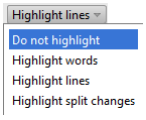
If the files have read-only status, they are not editable in the differences pane.

Diff viewer

Item	Tooltip and Shortcut	Description
	Previous Difference / Next Difference Shift+F7 F7	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to click the arrow buttons F7 / Shift+F7 once more and compare other files, depending on the Go to the next file after reaching last change option in the Differences Viewer settings . This behavior is supported only when the Differences Viewer is invoked from the Version Control tool window.
	Compare Previous/Next File Alt+Left Alt+Right	Click these buttons to compare the local copy of the previous/next file with its update from the server. Tip These buttons are available only when the Differences Viewer is invoked from the Version Control tool window.

	Go To Changed File Ctrl+N	Click this button to display the path to the compared file. Tip This button is available only when the Differences Viewer is invoked from the Version Control tool window.
	Jump to Source F4	Click this button to open the selected file in the active pane in the editor. The caret will be placed in the same position as in the Differences Viewer.
Viewer type		Use this drop-down list to choose the desired viewer type. The side-by-side viewer has two panels; the unified viewer has one panel only. Both types of viewers enable you to – Edit, for example, type text. Note that one can change text only in the right-hand part of the default viewer, or, in case of the unified viewer, in the lower ("after") line. – Perform the Apply/Append/Revert actions.

Whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

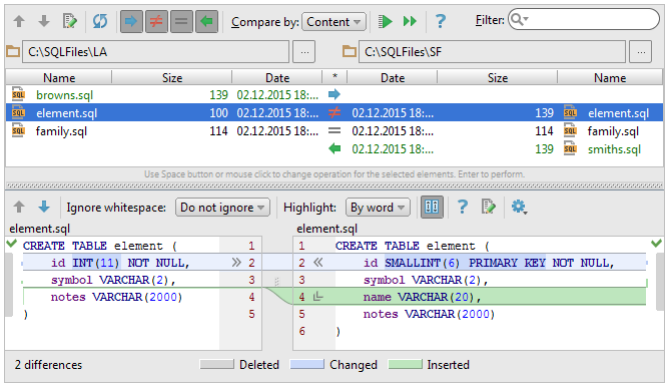
Highlighting mode		Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted – Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Collapse unchanged fragments	Click this button to collapse all unchanged fragments in both files. The amount of non-collapsible unchanged lines is configurable in the Diff & Merge settings page.
	Synchronize scrolling	Click this button to simultaneously scroll both differences panes; if this button is released, each of the panes can be scrolled independently.
	Editor settings	Click this button to reveal the drop-down list of available check commands. Select or clear these check commands to show or hide line numbers, indentation guides, white spaces, and soft wraps. Note that same commands are duplicated in the context menu of the differences viewer gutter. Right-click the gutter to show the menu of check commands.
	Include into commit Alt+I	This checkbox only appears if you invoke the Differences Viewer from the Commit Changes dialog with multiple changed files (all of which are deselected), and you explore the differences between them and hit the last difference in a file. Select this checkbox if you want to include the file you've reviewed into the commit.
	Move to Another Changelist F6	This button only appears if you invoke the Differences Viewer from the Commit Changes dialog with multiple changed files (all of which are deselected), and you explore the differences between them and hit the last difference in a file. Click this icon to move the file you've reviewed to another changelist.
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools settings page. This button only appears on the toolbar when the Use external diff tool option is enabled in the External Diff Tools settings page.
	Help F1	Click this button to show the corresponding help page.
	Ctrl+Tab	Use this keyboard shortcut to switch between the panes of the Differences viewer. The active pane has the cursor.
» «		Use these chevron buttons to apply differences between panes (in case of the side-by-side viewer) or between lines (in case of the unified viewer). The chevron buttons can change their behavior: – Click » « to apply changes. This behavior is the default one. – Press Ctrl to change » « to ⇟ or ⇞ and append changes.

Context menu

This menu appears on right-clicking an entry in the list of items. The commands listed below define actions to be taken for the selected entry and set the action icons in the * column.

Item	Icon	Description
Set Copy to		This command sets the specified icons in the * column to copy a file missing from one of the directories. If a file exists on one side, it will be copied; if it doesn't

Right/Left	exist, then the file on the other side will not be deleted.
Set Delete	This command sets the specified icons in the * column to delete file.
Set Do Nothing	Choose this command to remove an action icon.
Set Default	Choose this command to set the default action for the entry.
Warn When Delete	Select this option to display a warning when trying to delete a file that is located only in one of the two directories during their merge.



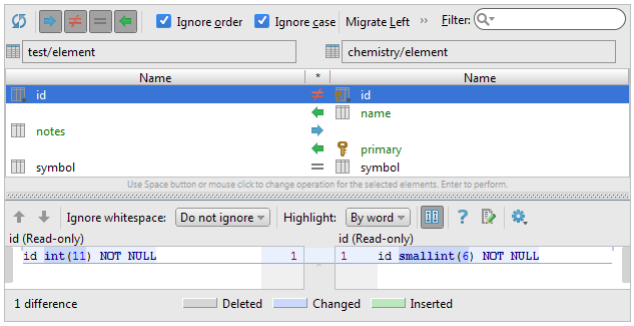
File comparison statuses and intended operations are shown in the column marked with an asterisk (*). To assign or change an operation, use the context menu associated with the corresponding cell. Alternatively, click the cell or press `Space` one or more times.

Applying operations:

- (`Enter`) applies the operations to selected files.
- (`Ctrl+Enter`) applies the operations to all the files.

Icon	Description
	The file exists only in the left-hand folder. The intended operation is to copy the file to the right-hand folder. If a file exists in both folders and you apply this operation, the file in the right-hand folder is overwritten.
	For the selected comparison criterion, the files are not identical. No operation is assumed. Study the file differences in the lower part of the view. You can choose to overwrite one of the files by assigning and applying the corresponding operation. You can as well modify the file contents. This may be done by typing or by using the following buttons and context menu commands: – or or Replace. Replace the fragment with the one from the other pane. – or or Insert. Insert the fragment into the other pane. – Remove. Remove the corresponding fragment. To undo the changes, use `Ctrl+Z`.
	For the selected comparison criterion, the files are identical. No operation is assumed.
	The file exists only in the right-hand folder. The intended operation is to copy the file to the left-hand folder. If a file exists in both folders and you apply this operation, the file in the left-hand folder is overwritten.
	Delete the file. The operation is not available for files that exist in both folders.

Differences viewer for table structures



Comparison statuses are shown in the column marked with an asterisk (*).

Icon	Description
	The column, constraint or index exists only in the left-hand table.
	The items exist in both tables but their definitions are different. You can study the differences in the lower part of the view.
	The items are identical.
	The item exists only in the right-hand table.

Generating ALTER TABLE statements

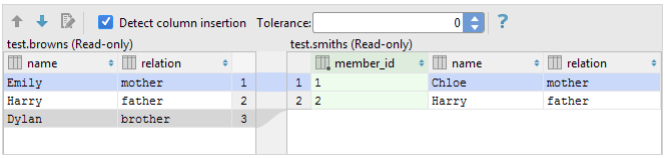
You can generate a set of `ALTER TABLE` statements for making data definitions in the left-hand and the right-hand parts identical. Use one of the following buttons in the upper part of the view:

- Migrate Left. The statements for the left-hand table or tables are generated.
- Migrate Right. This buttons does the same but for the right-hand table or tables.

The `ALTER TABLE` statements are generated for the items marked , and . If you don't want to generate the statements for some of those items, right click the * cell and select Set Do Nothing for all such items.

Differences viewer for tables

The primary purpose of the differences viewer for tables is to show the differences and similarities of data.



Docker Registry dialog

For this dialog to be available, the Docker integration plugin must be installed.

Specify your Docker image repository user account settings.

Item	Description
Name	The name for this set of settings.
Address	The image repository service URL, e.g. – registry.hub.docker.com for Docker Hub – quay.io for Quay
Username	The user name for your user account.
Password	Your password.
Email	The email address that you specified when creating your user account.
Server	The associated Docker Remote API configuration (used to connect to the service to check that your user account settings are correct).

Edit as Table: <file_name> Format dialog

Specify how your delimiter-separated values should be converted into table format.

When working on the conversion settings, use the table preview in the right-hand part of the dialog.

Item	Description
Formats	Select your file format and check the table preview. If you haven't achieved the desired result yet, adjust the settings. If you have changed the settings and want to save the changes, click this icon and select one of the following: − Save Changes. The settings are saved "under the same name", without creating a new format. (A format, in fact, is a named set of settings.) − Save As. The settings are saved "under a different name": a new format is created and you can specify the name for that new format.
Value separator	Select or type the character used for separating individual values.
Row separator	Select or type the character that should be treated as a row separator.
Null value text	The text to be used as a value if a cell contains `null` (an unknown value).
Add row prefix/suffix	Row prefix and suffix are character sequences which in addition to the row separator indicate the beginning and end of a row. If necessary, click the link and specify the row prefix and suffix in the fields that appear.
Quotation	Each line in the area under Quotation is a quotation pattern (see Quote values). A quotation pattern includes: − The left quotation character, the one inserted before a value. − The right quotation character, the one inserted after a value; usually, the same as the left quotation character. − An escape method or character for the cases when the quotation character is part of a value. E.g. Escape: duplicate means that if a quotation character occurs within a value, it is doubled. (You can specify your own escape character instead.) If there is more than one pattern, the first of the patterns is used. Use +, −, ↑ and ↓ to create, delete and reorder the patterns. To start editing an existing pattern, just click the pattern of interest.
Quote values	Specify in which cases the values should be quoted (i.e. enclosed within quotation characters). − When needed. A value is quoted only if it contains the value and/or the row separator. − Always. Any value is quoted in its text representation.
Trim whitespaces	If this check box is not selected, the Unicode whitespace characters that precede and follow the value separators are treated as parts of the corresponding values. If this check box is selected, the corresponding whitespace characters are ignored or removed.
First row is header	If this check box is selected, the first row is treated as containing column names. The settings that appear under Header Format have the same meanings as the ones above but are applied to the first row.
First column is header	If this check box is selected, the first column is treated as containing row names.

Edit Macros Dialog

Edit | Macros | Edit Macros

Use this dialog to change or delete the existing macros. Note that the dialog is not available, when there are no macros.

Item	Description
Existing macros list	The left-hand pane of the dialog shows the list of existing macros.
—	Click this button to delete a recorded macro.
	Click this button to change name of a recorded macro in the Rename Macro dialog.
Actions list	The right-hand pane shows the list of actions recorded for each macro. The list of actions for each macro is editable - you can remove unnecessary actions.
—	Click this button to delete an action from the macro.

Edit Project Path Mappings Dialog

The dialog box opens when you click ... next to the Path Mappings field in various cases (for example, in the Run/Debug Configuration: Node JS dialog , or when you configure a remote interpreter).

If in the current run/debug configuration you use an interpreter accessible through SFTP connection or located on a Vagrant instance, the mappings are automatically retrieved from the corresponding deployment configuration or `Vagrantfile` and listed in the dialog box. The mappings are read-only.

- To add a custom mapping, click + and specify the path in the project and the corresponding path on the remote runtime environment in the Local Path and Remote Path fields respectively. Type the paths manually or click ... and select the relevant files or folders in the dialog box that opens.
- To remove a custom mapping, select it in the list and click —.

Evaluate Expression

Run | Evaluate Expression

Editor Context Menu | Evaluate Expression

Alt+F8

Use this dialog box to calculate values of expressions during the debugging session.

Item	Description
Expression	Use this field to edit the expression to be evaluated. If an expression is selected in the editor, this field displays selection.

Export Threads

Run | Export Thread

Debug tool window | Threads | context menu | Export Threads

Alt+U+H

Use this dialog to export a thread to the specified text file.

Item	Description
Export to file	Type the target file to store the thread report.
...	Use this button to navigate to the desired location in the Select Path dialog.
Save	Click to save the current threads in a text file indicated in the Export to file field.
Copy	Click to copy the thread status to the Clipboard.
Cancel	Use to discard all changes and close the dialog.

Export to HTML

File | Export to HTML

Use this dialog to save selected files in HTML format.

Item	Description
File <name>	Click this radio button to print the file currently selected in the Project view, or open in the editor.
Selected text	Click this radio button to print the text selected in the editor.
All files in the directory	Click this radio button to print all files in the current directory.
Include subdirectories	Select this check box to print the files in the subdirectories of the current directory.
Output directory	Specify fully qualified path to the directory, where the resulting HTML file will be stored.
Show line numbers	Select this check box to include line numbers in the resulting HTML file.
Generate hyperlinks to classes	Select this check box to replace class names with the hyperlinks to the respective classes.
Open generated HTML in browser	Select this check box to show HTML file in the default browser after export.

File Cache Conflict

An external process has changed a file, opened and unsaved in PhpStorm, which results in two conflicting versions of a file. Resolve this conflict using the following options.

Item	Description
Load FS Changes	Click this button to load the file version produced outside of PhpStorm, and overwrite your local changes.
Keep Memory Changes	Click this button to preserve the version produced in PhpStorm and stored in cache.
Show Difference	Click this button to invoke the differences viewer that shows the version in the file system to the left, and PhpStorm version to the right. You can merge differences as required and load the desired version to PhpStorm. By default, the cache version is saved.

Find and Replace in Path

Edit | Find | Find in Path or Replace in Path

`Ctrl+Shift+F` or `Ctrl+Shift+R`

In the Project tool window : Find in Path or Replace in Path from the context menu for a directory

---

Specify what you want to find and where. In the Replace in Path dialog, also specify the replacement text or pattern.

Use `Ctrl+Shift+F` and `Ctrl+Shift+R` to switch between the find and replace modes.

Search pattern and replacement text

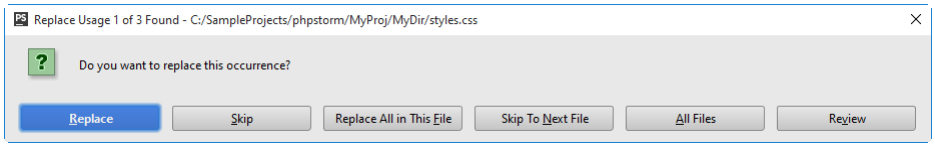
Item	Description
Text to find	In this field, specify the search pattern. Type the text manually or select one of the previously specified patterns from the drop-down list. – If you specify the search pattern through a regular expression, use the `$n` format in back references (to refer to a previously found and saved pattern). – This field can be left empty. If there is no text to find, but the File mask check box is selected, then search results include only the files matching the specified mask. Pressing `Alt+Down` in this field, will open a history drop-down list of your previous search entries.
Replace with	In this field, specify the replacement text. Type the text manually or select one of the previously specified substitutions from the drop-down list. – If you specify the replacement text through a regular expression, use the `$n` format in back references. – To use a backslash character `\` in a regular expression, escape the meaningful backslashes by inserting three extra backslashes in front: `\\\\` . This field is available only in the Replace in Path dialog.

Options

Use this tab to specify find and replace options.

Item	Description
General	In this area, specify additional search and replace parameters. The available options are: – Case sensitive - select this check box to have PhpStorm distinguish between upper and lowercase letters while searching. – Preserve case - if this check box is selected, PhpStorm retains the case of the first letter and the case of the initial string in general. For example, *MyTest* will be replaced with *Yourtest* if you specify *yourtest* as the replacement. This check box is disabled, if the Case sensitive or Regular expressions check box is selected. This field is available only in the Replace in Path dialog. – Whole words only (may be faster) - select this check box to have PhpStorm search for whole words only, that is, for character strings separated with spaces, tabs, punctuation, or special characters. This check box is disabled, if the Regular expressions check box is selected. – Regular expressions - select this check box if the specified search pattern should be treated as a regular expression . Tip Click the [Help] link next to the check box to view reference on regular expressions syntax. – Context - use this drop-down list to confine the search to a certain context, for example: – anywhere - select this option to search everywhere. – In comments - select this option to confine search to comments, ignoring the other occurrences. – In string literals - select this option to confine search to string literals, ignoring the other occurrences – Except... - select one of the exception options to perform search avoiding comments, string literals or both.
Scope	In this area, specify the scope to apply the search to. The available options are: – Whole project - select this option to search through the entire project. – Directory - select this option to perform search within the specified directory. By default, the text area already contains the directory name where a file currently opened in the editor is located (if you call the dialog from the editor), or where a file selected in the tool window is located (if you call the dialog from the tool window), or the directory name selected in the tool window. Pressing the ellipsis button opens the Select Path dialog, where you can select the necessary directory. – Recursively - this check box is only available for the directory search. If selected, sets the search to be performed in the chosen directory and its subdirectories. – Custom - select this option to search in a scope . You can choose one of the scopes from the drop-down list, or click the ellipsis button, and define a new scope in the Scopes dialog .
File name filter	In this area, specify additional settings to narrow down the search scope. – File mask - select this check box to narrow down the search scope through file masks. In the drop-down list, select the desired mask or specify a new one using wildcards. Wildcards can include: – `*` to substitute a set of any characters, – `?` to substitute a single character, – `!` to exclude files. Mind that `!` should go first in a particular file name pattern, for example, `!*.gant` You can specify multiple file masks, delimited with commas (for example, `*.xml,a?c.sql,!*.html`). Note also, that negated pattern (for example, `!*.min.js`) has implicit inclusion pattern `*` . This allows avoiding such constructs as `*, !*.min.js` for every file except minified javascript). If text to find is not entered , and this check box is selected, then PhpStorm find all files matching the specified mask, regardless of their contents.
Result options	Click the down arrow to reveal the result presentation options. – Skip results tab with one usage - Select this check box to be navigated directly to the found string in the editor, when only one usage is found. The check box is available only in the Find in Path dialog box. – Open in new tab - If selected, sets the search results to be displayed in a separate tab.
Find	– When you click this button in the Find dialog box, PhpStorm displays the encountered occurrences of the search string in the Find tool window , selects the first occurrence and opens the file with this occurrence in the editor and moves the focus to it.

- When you click this button in the Replace dialog box, PhpStorm displays the encountered occurrences of the search string in the Find tool window , selects the first occurrence and opens the file with this occurrence in the editor and moves the focus to it.
  At the same time, PhpStorm opens the Replace Usage dialog box, with the full path to the encountered occurrence in the title bar:



  Do one of the following:
  - To have the selected occurrence replaced, click Replace.
  - To preserve the selected occurrence and move to the next one, click Skip.
  - To have all the occurrences of the search string in the currently active tab replaced, click Replace All in This File.
  - To preserve the occurrences of the search string in the currently active tab (any) and move to the next file, click Skip to Next File.
  - To have all the detected occurrences replaced, click All Files.
  - To switch to the manual mode, click Preview. The Replace Usage dialog box closes and the focus moves to the Find tool window. Do one of the following:
    - Browse through the list of detected occurrences, select the ones you want to replace and then click Replace Selected.
    - To have all the occurrences changed click Replace All.

Preview

Use this tab to check the preview of your search target.

You can press `Up` or `Down` keys to navigate between entries in the preview area without leaving the Text to find field. You also can press `F4` to get to the selected entry in the editor.

Find Usages Dialogs

In this section:

- Find Usages
- Find Usages. Class Options
- Find Usages. Method Options
- Find Usages. Variable Options

Find Usages

Edit | Find | Find Usages Settings

Alt+F7

The dialog also opens when you click in the Show Usages pop-up window which lists all the occurrences of the symbol at caret.

Use this dialog to configure the search procedure and scope when looking for occurrences of the following data:

- Fields, variables and parameters
- Classes, tags, attributes, and references in the HTML, XML, and CSS files
- Symbols at caret.

The search results are displayed in the Find tool window .

Item	Description
Skip results tab with one usage	Select this check box to be navigated directly to the found usage without the Find tool window displayed, when only one usage is found.
Scope	Specify the search scope . Select a pre-defined scope from the drop-down list or click ... to define a custom scope in the Scopes dialog .
Open in new tab	Select this check box to have the results of each search shown on a separate tab of the Find tool window. If the check box is not selected, the search results will be shown on the current tab.

Find Usages. Class Options

Edit | Find | Find Usages

Alt+F7

This section describes the controls for specifying Class Usage Search options in the Find Usages dialog box.

The dialog opens when you click in the Show Usages pop-up window which lists all the occurrences of the symbol at caret.

Item	Description
Scope	In this area, specify the scope of search. Select a pre-defined scope from the drop-down list or click the Browse button ... to open the Scopes dialog box, where you can define a custom scope.
Open in new tab	Select this check box to have the results of each search shown in a separate tab of the Find Results window. If the check box is cleared, the search results will overwrite the contents of the current tab.

Find Usages. Method Options

Edit | Find | Find Usages

`Alt+F7`

This section describes the controls for specifying Method Usage Search options in the Find Usages dialog box.

The dialog opens when you click in the Show Usages pop-up window which lists all the occurrences of the symbol at caret.

Item	Description
Search for text occurrences	Select this check box to have text contents and comments involved in searching.
Skip results tab with one usage	Select this check box to be navigated directly to the found usage without the Find tool window displayed, when only one usage is found.
Scope	In this area, specify the scope of search. Select a pre-defined scope from the drop-down list or click the Browse button ... to open the Scopes dialog box, where you can define a custom scope.
Open in new tab	Select this check box have the results of each search shown in a separate tab of the Find Results window. If the check box is cleared, the search results will overwrite the contents of the current tab.

Find Usages. Variable Options

Edit | Find | Find Usages

Alt+F7

This section describes Field, Variable, or Parameter usage search options in the Find Usages dialog.

The dialog opens when you click in the Show Usages pop-up window which lists all the occurrences of the symbol at caret.

Item	Description
Skip results tab with one usage	Select this check box to be navigated directly to the found usage without the Find tool window displayed, when only one usage is found.
Scope	In this area, specify the scope of search. Select a pre-defined scope from the drop-down list or click the Browse button ... to open the Scopes dialog box, where you can define a custom scope.
Open in new tab	Select this check box have the results of each search shown in a separate tab of the Find Results window. If the check box is cleared, the search results will overwrite the contents of the current tab.

Generate Instance Document from Schema Dialog

Tools | XML Actions | Generate XML Document from XSD Schema

In this dialog box, specify the options for generating an XML file based on the XSD (XML Schema Definition) Schema that describes its structure.

The menu item and the dialog box are available when the file opened in the active editor tab contains an XML Schema.

Item	Description
Schema path	In this field, specify the location of the `.xsd` Schema file to be used as the basis for XML document generation. By default, the field shows the full path to the current file. To use another Schema, click the Browse button ... and select the desired file in the dialog that opens .
Instance document name	In this text box, specify the name of the output XML file to be generated. By default, the generated XML file will inherit the name of the source Schema and will have the `.xml` extension. If you type another name for the document to be generated, make sure the extension is correct. The default location for the generated document is the directory where the source `.xsd` Schema file resides. To specify another location, click the Browse button ... and select the desired path in the dialog that opens .
Element name	In this drop-down list, specify the local name of the global element to be used as the root of the generated document.
Enable restriction check	Select this check box to have PhpStorm take restriction particles into consideration (if the specified Schema uses any).
Enable unique check	Select this check box to have PhpStorm take uniqueness particles into consideration (if the specified Schema uses any).
Status	View the information in this read-only field to track and improve discrepancies when configuring the generation procedure.

Generate Schema from Instance Document Dialog

Tools | XML Actions | Generate XSD Schema from XML File

In this dialog box, specify the options for generating an XSD (XML Schema Definition) Schema that describes the structure of the desired XML file.

The menu item and the dialog box are available when an XML document is opened in the active editor tab.

Item	Description
Instance document path	In this text box, specify the full path to the XML document to be used as the basis for Schema generation. By default, the field shows the full path to the current file. To use another XML document, click the Browse button ... and select the desired file in the dialog that opens .
Result Schema file name	In this text box, specify the name of the output file for the Schema to be generated. By default, the generated XSD Schema file will inherit the name of the instance document used and will have the `.xsd` extension. If you type another name for the Schema to be generated, make sure the extension is correct. The default location for the generated Schema is the directory where the source XML instance document resides. To specify another location, click the Browse button ... and select the desired path in the dialog that opens .
Design type	Use this drop-down to specify how to declare elements and complex types. The available options are: − Local elements / global complex types − Local elements / types − Global elements / local types
Detect simple content types	From this drop-down list, choose the type to render leaf text, which should be distinguished from the text used. The available options are: − String − Smart
Detect enumeration limit	In this text box, type the number of occurrences to cause the Schema enumeration appearance. To suppress Schema enumeration, specify 0.

Incoming Connection Dialog

The dialog is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

The dialog box opens when you initiate a Zero-Configuration debugging session without previously configuring a Debug Server with path mappings, so the debugger cannot set correspondence between the scripts in the running application and the files in your PhpStorm project.

The upper psrt of the dialog box is read-only and shows the host and port where the script is executed, the path to the script relative to the server document root, and the absolute path to the script on the server.

In this dialog box, either specify the path mappings manually or import them from the relevant deployment configuration. Learn more at Creating a PHP Debug Server Configuration , Configuring Synchronization with a Web Server , and Zero-Configuration Debugging .

To continue debugging with the imported or manually specified configuration settings, click Accept.

Item	Description
Import mappings from deployment	Choose this option to use the mappings specified in a server access configuration (deployment configuration.). – Project: this read-only field shows the name of the current PhpStorm project. – Deployment: from this drop-down list, choose the deployment configuration to import the mappings from. If PhpStorm detects a deployment configuration which seems relevant, the configuration is preselected in the Deployment drop-down list. If PhpStorm does not detect a relevant configuration: 1. Choose the most suitable configuration from the drop-down list or click ... and create a new configuration in the Deployment dialog box that opens, whereupon the new configuration is added to the list. 2. In the Deployment root text box, type the absolute path to the server root folder – Preview: this area shows the absolute path to the project file which corresponds to the currently executed script according to the mappings from the selected configuration.
Manually choose local file or project	When you select this option, PhpStorm displays the project tree view where you can select a project file and map the currently executed script to it. You can also select and map the entire project.

Map External Resource Dialog

Specify a URI and select a local XSD or DTD file for the specified URI. (If an XML file references the specified URI, it's validated according to a selected local file.)

Item	Description
URI	Use this field to edit the URI. This may be a URL of an XSD or DTD file (e.g. `http://www.example.org/xsds/example.xsd`) or a namespace URI (e.g. `http://www.example.org`).
Schemas	This tab lists XSD and DTD files available in your project, libraries, SDKs and PhpStorm installation.
Explorer	This tab implements a file browser .

Non-Project Files Protection Dialog

This dialog appears when you try to edit non-project files (e.g, library sources, external sources etc.), and protects them from accidental modifications.

Item	Description
These files do not belong to the project	This area displays a non-project file that you are trying to edit.
I want to edit this file anyway	Select this option to disable protection for the listed files.
I want to edit all files in this directory	Select this option to disable protection for the listed files and all files in the same directory.
I want to edit any non-project file in the current session	Select this option to disable protection completely.

All options are effective during current session, once the IDE restarted, protection will be reenabled.

New Project From Existing Files Wizard

File | New Project From Existing Files

Use this Wizard to set up new projects around existing files that reside locally or on remote hosts.

The wizard gives you a way to process specific files from a distributed project without downloading or copying it entirely. PhpStorm creates a project directory in the specified location with the `.idea` folder that contains the project files .

Choose Your Scenario

On the first page of the Wizard, choose the way to access the files around which a new project will be set up.

Item	Description
Web server is installed locally, source files are located under its document root.	Select this option if you have a local Web server and you want to set up a project around existing files or folders that are located below the folder appointed as the server document root, for example below `.\htdocs` . Clicking Next brings you to the Create New Project: Choose Project Directory page.
Web server is installed locally, source files are located elsewhere locally.	Select this option if you have a local Web server and the files or folders you want to set up a project around are also stored locally but not under the folder appointed as the server document root. Clicking Next brings you to he Create New Project: Specify Local Path page.
Web server is on a remote host, files are accessible via network share or mounted drive.	Select this option to have PhpStorm copy files to a local drive via network and set up a project around them. Clicking Next brings you to he Create New Project: Specify Local Path page.
Web server is on a remote host, files are accessible via FTP/SFTP/FTPS.	Select this option to have PhpStorm download files from a remote server via the FTP, SFTP, or FTPS protocol. Clicking Next brings you to he Create New Project: Specify Local Path page.
Source files are in a local directory, no Web server is yet configured.	Select this option if you want to work with files in a certain local directory without using any Web server. Clicking Next brings you to the Create New Project: Choose Project Directory page.

Create New Project: Choose Project Directory

File | New Project From Existing Files - Web server is installed locally, source files are located under its document root

File | New Project From Existing Files - Source files are in a local directory, no Web server is yet configured

---

On this page of the wizard, specify the location of the files, appoint the project root, and configure the structure of the project to be created. The project root is the folder which is the parent for all the project sources. By default, all subfolders in this folder are treated as sources and their files are involved in indexing, searching, parsing, code completion, etc. Use the icons on the toolbar ro change this status.

If your sources are located in several independent parent folders, create a project in one of these folders and configure other sources as additional content roots on the Directories page of the Settings dialog box.

For more details about configuring project contents, see Content Root and Configuring Folders within a Content Root .

Item	Description
Project root	Select the folder which is the parent for all the project sources and click this icon or choose Project Root on the context menu of the selection. PhpStorm marks the selected folder with the icon and adds the `.idea` directory with settings to it.
Test Sources	Click this icon to have PhpStorm treat the contents of tje selected folder as related to unit testing. Alternatively, choose Resource Root on the context menu of the selection.
Resource Root	Click this icon to enable PhpStorm to complete relative paths to resources under the selected folder. This means that the contents of the selected folder can be referenced relative to it instead of specifying full paths. Alternatively, choose Resource Root on the context menu of the selection.
Excluded	Click this button to have PhpStorm ignore the selected directory during indexing, parsing, code completion, etc. Alternatively, choose Excluded on the context menu of the selection.

Create New Project: Specify Local Path

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - Web server is on a remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - Web server is on a remote host, files are accessible via FTP/SFTP

On this page, specify the name of the project to be set up and the local folder to download the project sources to.

Item	Description
Project name	In this text box, type the name of the project to be set up based on the downloaded files.
Project local path	In this text box, specify the local folder where the new project will be set up. First specify the parent folder where the project folder will be created. Type the path manually or click the Browse button ... and select the desired location in the dialog that opens . As you type the project name, it is added to the project path.
Deployment options	In this area, specify the download configuration to use. The available options are: − Default - click this option to have PhpStorm download the sources according to the settings specified in the Options dialog box on the Deployment page. − Custom - click this option to customize the default settings in the Create New Project: Review Deployment Options dialog box that opens when you click Next.

Create New Project: Review Deployment Options

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - Web server is on a remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - Web server is on a remote host, files are accessible via FTP/SFTP/FTPS

The page opens upon clicking Next on the Create New Project: Specify Local Path page of the wizard if you have selected the Custom option in the Deployment options area. Use this page to customize the default deployment procedure to the current download.

Item	Description
Exclude items by name	In this text box, specify patterns for the names of files and folders that you do not need to be deployed. Use semicolons as delimiters. Wildcards are welcome. The exclusion is applied recursively. This means that if a matching folder has subfolders, the contents of these subfolders are not deployed either.
Operations logging	Use this drop-down list to specify how much detailed logging you need to have. The available options are: – Errors only: select this option to have the log show only errors occurred during upload. – Brief: select this option to have all events reflected in the log but without details. – Detailed: select this option to have more details on the upload shown in the log, for example, full file paths.
Stop operation on the first error	Select this check box to have data transfer stopped as soon as an error occurs.
Overwrite up-to-date files	Do one of the following: – Select this check box to have all the files uploaded no matter whether they have been changed since the previous upload or not. – Clear the check box to upload only files that have been changed since the previous upload.
Preserve files timestamps	Select this check box to prevent resetting timestamps of files on upload.
Delete target items when source ones do not exist	If this check box is selected, any file in the destination directory will be removed if the file with this name is not involved in the current upload. This option is applicable when synchronization if performed from the Project tool window or from the Remote Host tool window.
Create empty directories	Select this check box to have an empty directory on the server created automatically if a new local directory has been created in your project since the last upload in the source folder.
Prompt when overwriting or deleting local items	Select this check box to have PhpStorm ask you for confirmation before overwriting or deleting local items for synchronization during download.
Upload changed files automatically to the default server	From this drop-down list, choose when you want PhpStorm to automatically upload a file to the default server. The available options are: – Always: choose this option to have a file uploaded upon each save, no matter automatic or explicitly invoked. – On explicit save action: choose this option to have a file uploaded after save only if this save was invoked manually by choosing File | Save all or pressing `Ctrl+S`. – Never: choose this option to suppress automatic upload. The default server configuration is appointed on the Deployment page by selecting the desired configuration in the list and clicking the Use as Default toolbar button.
Upload external changes	Select this check box to have PhpStorm upload also the local changes that were made using a third-party tool.
Override default permissions on files	Select this check box to change the default permissions assigned to uploaded files on remote hosts. Click the Browse button ... to open the Files Default Permissions dialog box, where you can manage access to uploaded files on remote hosts by assigning permissions.
Override default permissions on folders	Select this check box to change the default permissions assigned to uploaded folders on remote hosts. Click the Browse button ... to open the Folders Default Permissions dialog box, where you can manage access to uploaded folders on remote hosts by assigning permissions.
Warn when uploading over newer file	Use this drop-down list to define the version-control policy to apply when uploading files to remote hosts. Depending on this choice, PhpStorm either checks whether any changes have been made to the corresponding files on the remote host since you downloaded them or just overwrites the remote files. – No: choose this option to have the file on the remote host overwritten with its local copy silently. All the changes made to the remote file since your last synchronization will be abandoned. – Compare timestamp and size: if you choose this option, PhpStorm performs two checks: 1. Compares the sizes of the local and remote files. 2. Compares the remote file timestamp set at the moment of the last synchronization with the current remote file timestamp. If the files differ in their size or the remote file timestamps differ, PhpStorm opens a Difference Viewer for Files , where you can explore and integrate the differences. This type of check depends on the timezone setting. If the timezone setting on your local machine is different from that on the remote host, the check may be successful even though the file versions actually differ. – Compare content: when this option is chosen, PhpStorm compares the content of the local and remote files. If any diversions are detected, PhpStorm opens a Difference Viewer for Files , where you can explore and integrate the differences.
Notify about remote changes	Select this check box to receive notifications about changes on the remote host. The check box is available only when the Compare timestamp and size: or Compare content: option is selected in the Warn when uploading over newer file drop-down list.
SFTP advanced options (IDE level)	
Add new host key to known_hosts	Choose whether PhpStorm should ask about connecting to a host not mentioned in the file `known_hosts`. The following options are available: – Always: always connect and add its record to the file `known_hosts`. – Ask: ask whether PhpStorm should connect to a host and add its record to the file `known_hosts` – Never: do not connect.
Hash hosts in known_hosts file	If this check box is selected, the new host record will be stored in hash format.

Create New Project: Specify Local Server

File | New Project From Existing Files - Web server is installed locally, source files are located under its document root

The page opens when you click Next on the Create New Project: Choose Project Directory page. On this page of the wizard, choose the local server access configuration to use. You can select an existing configuration or define a new one.

Item	Description
Add new local server	Select this option to move to the Create New Project: Add Local Server page and specify a new Web server configuration.
Use existing server	Select this option to use one of the existing Web server configurations from the list below.
Name	This read-only field shows the name of the local server access configuration.
URL	This read-only field shows the URL address of the server configuration root (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.).
Don't check HTTP connection to server	– When this check box is cleared, PhpStorm checks whether the specified URL address ensures successful connection to the server. – When this check box is selected, PhpStorm moves to the next page of the Wizard without any connection check.

Create New Project: Add Local Server

File | New Project From Existing Files - Web server is installed locally, source files are located under its document root

The page opens if you have selected the Add new local server option on the Create New Project: Specify Local Server page. On this page, define a new local Web server access configuration.

Item	Description
Name	In this text box, type the name of the new configuration.
Web server root URL	In this text box, type the URL address of the server configuration root (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.). Both the HTTP and the HTTPS protocols are supported. To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_name>.cert>
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`.
   Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

Open	Click this button to open the page at the specified URL address in the default browser and make sure that the URL actually points at the server configuration root.

Create New Project: Specify Remote Server

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - My web server is on remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - My web server is on remote host, files are accessible via FTP/SFTP/FTPS

The page opens when you click Next on the Create New Project: Specify Local Path page. On this page of the Wizard, choose the remote Web server to use. You can select a configured Web server or define a new configuration.

Item	Description
Add new remote server	Select this option to move to the Create New Project: Add Remote Server page and specify a new Web server configuration.
Use existing server	Select this option to use one of the existing Web server configurations from the list below.
Name	This read-only field shows the name of the remote server configuration.
URL address	This read-only field shows the URL address of the server document root according to the chosen configuration.
Mounted Folder	This read-only field shows the path to the mounted drive. The field is available if you have selected the My web server is on remote host, files are accessible via network share or mounted folder option on the first page of the Wizard.
Host	This read-only field shows the URL address of the FTP, SFTP, or FTPS host in the format `ftp://<host_name>:<port>` or `ftp://<host_name>:<port>` Note The field is available if you have selected the My web server is on remote host, files are accessible via FTP/SFTP/FTPS option on the first page of the Wizard.
Don't check HTTP connection to server	− When this check box is cleared, PhpStorm checks whether the specified URL address ensures successful connection to the server. − When this check box is selected, PhpStorm moves to the next page of the Wizard without any connection check.

Create New Project: Add Remote Server

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - My web server is on remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - My web server is on remote host, files are accessible via FTP/SFTP/FTPS

The page opens if you have selected the Add new remote server option on the Create New Project: Specify Remote Server page. On this page, specify a new configuration of settings to connect and authenticate to a remote server.

Item	Description	Available for
Name	In this text box, type the name of the new remote server configuration.	
Access type	From this drop-down list, choose the way to access the server. Use the Up and Down keyboard keys to scroll through the list of server configuration types. The available options are: − FTP: choose this option to have PhpStorm access the server via the FTP file transfer protocol . − SFTP: choose this option to have PhpStorm access the server via the SFTP file transfer protocol. − FTPS: choose this option to have PhpStorm access the server via the FTP file transfer protocol over SSL (the FTPS extension). − Local or mounted folder: choose this option if the Web server is running in a local or a mounted folder and its document root is NOT the parent of the project root. − In-place: choose this option if the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.	
FTP/SFTP/FTPS host	In this text box, specify the host name of the FTP/SFTP/FTPS server to download the files from.	
Port	In this text box, specify the port to use. The default values are: − 21 for FTP and FTPS − 22 for SFTP	
User name	In this text box, type your user name for authentication to the server.	
Log in as anonymous	Select this check box to enable anonymous access to the server with your email address as password.	
Auth type	From this drop-down list, select the client authentication method. The available options are: − Password - select this option to use standard authentication through a password. − Key pair (OpenSSH) - select this option to use SSH authentication via a key pair. Warning! PhpStorm supports private keys generated in accordance with the OpenSSH protocol .	
Password	In this text box, type your password for authentication to the server.	
Private key file	In this text box, specify the location of your private key file.	
Passphrase	In this text box, specify your authentication passphrase.	
Save password	Select this check box to have PhpStorm remember the specified password.	
Save passphrase	Select this check box to have PhpStorm remember the specified passphrase.	
Test FTP/SFTP/FTPS connection	Click this button to check that the specified settings ensure successful connection via FTP/SFTP/FTPS.	
Root path	In this text box, specify the server configuration root relative to your user home which was defined when you registered your account. This folder will be the highest one in the folder structure accessible through the current server configuration. Do one of the following: − Accept the default value `/` , which points at the user home folder on the server. − Type the path manually. − Click the Browse button ... and select the desired folder in the Choose Root Path dialog box that opens. − Click the Autodetect button and have PhpStorm detect the user home folder settings on the FTP/SFTP server and set up the root path according to them. The button is only enabled when you have specified your user name and password.	
Autodetect	Click this button to have PhpStorm detect the user home folder settings on the FTP/SFTP/FTPS server and set up the root path according to them.	
Explicit	Choose this option to have the explicit (active) security applied. Immediately after establishing connection, the FTP client on your machine sends a command to the server to establish secure control connection through the default FTP port.	
Implicit	Choose this option to have the implicit (passive) security applied. In this case, security is provided automatically upon establishing connection to the server which appoints a separate port for secure connections.	
Advanced options	Click this button to specify the following additional uploading settings in the Advanced Options dialog box that opens. The set of controls in the dialog box depends on the server access type, for details, see Advanced Options Dialog .	
Web server root URL	In this text box, specify the URL address of the Web server root folder . Both the HTTP and the HTTPS protocols are supported. To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle	

JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_name>.cert>
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`. Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

Open

Click this button to make sure that the specified server root URL address is accessible and points at the correct Web page.

Don't check HTTP connection to server

– When this check box is cleared, PhpStorm checks whether the specified URL address ensures successful connection to the server. Both the HTTP and the HTTPS protocols are supported.
To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracl
JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_name>.cert
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`. Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

– When this check box is selected, PhpStorm moves to the next page of the Wizard without any connection check.

Create New Project: Choose Remote Path

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - My web server is on remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - My web server is on remote host, files are accessible via FTP/SFTP/FTPS

The page opens when you click the Next button on the Create New Project: Specify Remote Server or Create New Project: Add Remote Server page.

The page shows a tree view of folders under the server configuration root (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.) On this page, appoint the folder that will become the project root .

If applicable, configure folders under the project root by selecting the relevant folders and clicking the toolbar buttons. Alternatively, you can do it later in the Directories page of the Settings / Preferences Dialog .

ItemTooltip	Description	
	Project Root	Click this button to make the selected folder the project root .
Please note the following: − After you specify the project root, all its subfolders are displayed on the green background. − The other toolbar buttons (Excluded from Download , Resource Root , and Excluded from Indexing) also become enabled only after the project root is appointed.		
	Excluded from Download	Click this button to have PhpStorm skip the selected folder under the project during download. These can be sources that you should not update at all, media, caches, temporary files, that you actually do not need in your work. Exclusion is applied recursively. This means that if the selected folder contains subfolders they are automatically marked as excluded as well.
	Resource Root	Click this button to enable PhpStorm to complete relative paths to resources under the selected folder.
	Excluded from Indexing	Click this button to have PhpStorm completely ignore the selected directory during indexing, parsing, code completion, etc.

Create New Project: Specify Web Path (Remote)

File | New Project From Existing Files - Web server is installed locally, source files are located elsewhere locally

File | New Project From Existing Files - My web server is on remote host, files are accessible via network share or mounted drive

File | New Project From Existing Files - My web server is on remote host, files are accessible via FTP/SFTP/FTPS

The page opens when you click Next on the Create New Project: Choose Remote Path page. On this page, compose the Web path to access the project root directory via HTTP.

Item	Description
Web path for project root	In this text box, specify the path to the selected project directory relative to the Web server document root URL .
Project URL	This read-only field shows the full URL address via which the project root directory will be available. By default, the field is filled in with the name of the selected project root directory. PhpStorm dynamically constructs the full URL address as you type the relative path in the Web path for project root text box.

Create New Project: Specify Web Path (Local)

File | New Project From Existing Files - Web server is installed locally, source files are located under its document root

The page opens when you click Next on the Create New Project: Specify Local Server or on the Create New Project: Add Local Server page. On this page, compose the Web path to access the project root directory via HTTP.

Item	Description
Web path for project root	In this text box, specify the path to the selected project directory relative to the URL of the server configuration root URL (The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.).
Project URL	This read-only field shows the full URL address at which the project root directory will be available. By default, the field is filled in with the name of the selected project root directory. PhpStorm dynamically constructs the full URL address as you type the relative path in the Web path for project root text box.
Review PHP settings	Select this check box to move to the Create New Project: PHP Settings page and check the PHP installation.

Create New Project: PHP Settings

File | New Project From Existing Files - Web server is installed locally, source files are located under its document root

The page opens if you have selected the Review PHP settings check box on the Create New Project: Specify Web Path (Local) page of the Wizard. Use this page to ensure that a PHP engine is installed on the specified local Web server and to configure a list of include paths. The page shows the PHP engine version installed on the selected local Web server. If no PHP engine is detected, PhpStorm displays a corresponding warning.

Item	Description
Include paths on server	Use the controls in this area to manage a list of include paths on the server: – Add - click this button to have a new entry added to the list. – Remove - click this button to remove the selected entry from the list. – Browse ( ) - click this button to select the desired folder in the corresponding dialog .

Open Task Dialog

Tools | Task | Open Task

Tools | Task | Switch Task - Open Task

Tasks combo on the main toolbar | Open Task

`Shift+Alt+T`

Use this dialog box to create a new task or specify the name of an existing task to open. Optionally, specify whether the PhpStorm's behavior related to opening or creating a task, changes.

On this page:

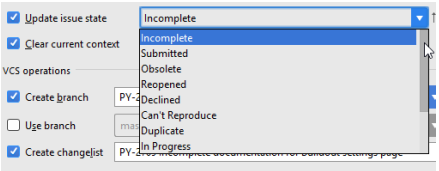
– Suggestion list

– Open Task dialog

Suggestion list

Item	Description
Enter task name	In this text box, type the name of the new task or the name of an existing task to open.
Include closed tasks	Select this check box to include closed tasks into the suggestion list.
	Click this button to open the Servers page of the Settings dialog box.

Open Task dialog

Item	Description
Open task: <task name>	This read-only field shows the name of the task selected from the suggestion list .
Update issue state	Select this check box to choose the task state from the drop-down list:  If this check box is not selected, the drop-down list is disabled.
Clear current context	Select this check box to have the context associated with the current task cleared, when the new/selected task is opened.
VCS operations	

Note This section only appears in the dialog box, when version control is enabled in your project.

Create branch <branch name> from <base branch name>	Select this check box to create a new branch from the existing one.
Use branch	Select this check box to use the existing branch.
Create changelist	Select this check box to have PhpStorm create a new changelist for the specified task. By default, the text field shows the issue tracker item's description.

Optimize Imports Dialog

Code | Optimize Imports

`Ctrl+Alt+O`

Note that this dialog box appears when you select a Optimize Imports action on the directory.

In this dialog box, specify where you want PhpStorm to remove unused import statements from, in order to optimize the import procedure.

Item	Description
Only VCS changed files	If this check box is selected, then reformatting will apply only to the files that have been changed locally , but not yet checked in to the repository. This check box is only available for the files under version control.

Override Server Path Mappings Dialog

The dialog box opens when you click the Server paths mappings button in the Debug area of the Run/Debug Configuration: PHP Web Application or Run/Debug Configuration: PHPUnit by HTTP dialog box.

Use this dialog box to map folders on your local machine with folders on the server. These mappings are used during remote debugging and testing as the basis for switching between a test or a stack trace of an exception or assertion and the corresponding source code.

Tip PhpStorm detects these mappings automatically but still provides you with this possibility to specify them manually.

Item	Description
Local Path on Client	In this text box, specify the absolute path to the desired local folder. Type the path manually or click the Browse button and select the desired folder in the dialog that opens .
Local Path on Server	In this text box, specify the path to the corresponding folder on the server according to the file system used on the server.
Add	Click this button to have a new line added to the list of mappings.
Remove	Click this button to remove the selected mapping from the list.

PhpStorm License Activation Dialog

To open this dialog: Register on the Welcome screen or Help | Register.

You can evaluate PhpStorm Ultimate for 30 days. After that period, you need to buy PhpStorm and activate your license.

The upper part of the dialog reflects your PhpStorm usage status (e.g. Free evaluation) and, if appropriate, provides related controls (e.g. Buy PhpStorm).

The license activation options are in the lower part of the dialog under Activate new license via.

- PhpStorm usage status-related controls
- License activation options

PhpStorm usage status-related controls

Item	Description
Buy PhpStorm	Click this button to go to the JetBrains Web site to study the PhpStorm purchasing options and to buy a license.
Evaluate for free for 30 days	Click this button to start evaluating PhpStorm.

License activation options

Item	Description
License key	Select this option if you have a license key. Specify your user name (or company name) and the key. (The corresponding information is in your license certificate email.) The recommended way of entering the name and the key is by copying the information from the certificate email and then pasting it into the fields (`Ctrl+C` / `Ctrl+V`). If PhpStorm rejects the license information, see The key is not accepted for a possible solution.
License server	Select this option if there is the PhpStorm License Server on your company's intranet. Specify the server URL in the License server address field. For the address to be entered automatically, click the Discover server button. To find out if your company is using the License Server and what its URL should be, contact your system administrator.
JetBrains Account	Select this option if your PhpStorm license is linked with your JetBrains Account . Specify your JetBrains Account access credentials.

Manage Project Templates Dialog

Tools | Manage Project Templates

The upper part of the dialog box shows a list of your project templates. When you select a template, its description is shown in the bottom area of the dialog box.

To delete an unnecessary template, select it and click — or press `Alt+Delete`.

Print

File | Print

Use this dialog box to specify the text to print and configure the print layout.

In this section:

- Scope
- Settings
- Header and Footer
- Advanced

Scope

Item	Description
File <name>	Select this option to print the file, which is currently selected in the Project view or opened in the editor.
Selected text	Select this option to print the text selected in the editor.
All files in the directory	Select this option to print all files in the current directory.
Include subdirectories	Select this check box to have the files in the subdirectories of the current directory printed as well.

Settings

In this tab, specify the basic print layout settings.

Item	Description
Paper size	From this drop-down list, select the desired paper size.
Font	From the drop-down lists in this area, select the desired font style and size.
Show line numbers	Select this check box to have line numbers printed.
Draw border	Select this check box to have a border printed.
Orientation	In this area, specify the paper orientation. The available options are: – Landscape – Portrait
Style	In this area, specify the style of the printout by selecting the relevant check boxes. The available options are: – Color printing – Syntax printing – Print as graphics

Header and Footer

In this tab, specify the contents and placement of the header and footer.

Item	Description
Text line	In this text box, specify the contents of the header or footer. If necessary, combine plain text with print keywords. By default, PhpStorm suggests to print the name of a file `$FILE$` in the header and the current page number `$PAGE$` of all pages `$TOTALPAGES$` in the footer. The following print keywords are recognized: – `$FILE$` prints fully qualified file name. – `$PAGE$` – `$DATE$` – `$TIME$` – `$FILENAME$` prints file name without path. – `$TOTALPAGES$`
Placement	Use this drop-down list to specify whether the above line will be printed in the header or in the footer.
Alignment	From this drop-down list, select the desired alignment.
Font	In this area, specify the desired font style and size to print the header and footer text.

Advanced

Item	Description
Wrapping	In this area, configure text wrapping. The available options are: – No wrap – Wrap at word breaks
Margins	Use the text boxes in this area to specify the margins in inches.

Productivity Guide

Help | Productivity Guide

The Productivity Guide shows usage statistics for PhpStorm features.

The table in the upper part lists the features. Select a feature to see its description in the lower part.

To sort the information, click a cell in the header row.

Item	Description
Feature	The name of the feature.
Group	The group to which the feature belongs.
Used	How many times you used the feature.
Last used	When the feature was last used.

PSI Viewer

Tools | View PSI Structure

Use this viewer to explore internal structure of the various files or fragments of source code, as they are interpreted by PhpStorm. The dialog box is non-modal and enables you to keep on working with PhpStorm while being opened.

Item	Description
Show PSI structure for	Use this drop-down list to specify file type, or language constructs to be explored. The set of recognized file types depends on the supported languages and installed plugins.
Show PsiWhiteSpace	If this check box is selected, the generated tree view will contain `PsiWhiteSpace` nodes, corresponding to the spaces in the source code. When you select of clear this check box, the tree view of the PSI structure changes accordingly.
Show Tree Nodes	
Dialect	This drop down list becomes available for the languages that support dialects, for example, SQL, JavaScript etc.
Text	Use this pane to enter source code to be explored. PhpStorm suggests the following ways to supply code: – Type immediately within the text area. – Paste from clipboard. If you have copied some text from the editor, and then open the PSI viewer, the previous contents of the Text pane is selected, which enables you to overwrite it from the clipboard using `Ctrl+V`, or `Ctrl+Shift+V`. Note that some editing features are also available: removing line at caret `Ctrl+Y`, duplicating text `Ctrl+D`, and adding line with `Shift+Enter`.
PSI Structure	This read-only pane displays the PSI structure tree view, generated on clicking the Build PSI Tree button, according to the file type selected in the Show PSI structure for drop-down list. Navigating though the tree view highlights the corresponding fragments of source code in the Text pane. If currently selected tree node has references, they are also displayed in the References pane.
References	This read-only field shows references to the nodes of the PSI Structure tree view (if any). Unresolved references are shown red; the corresponding fragments of source code are also highlighted with a red frame.
Build PSI Tree	Click this button to generate PSI structure tree view of the code in Text pane, according to the file type selected in the Show PSI structure for drop-down list. If source code in the Text pane has been modified, use this button to refresh the tree view.

Pull Image dialog

For this dialog to be available, the Docker integration plugin must be installed.

Specify the settings for pulling an image from a Docker image repository such as Docker Hub or Quay .

Item	Description
Registry	The URL of the image repository service or a Docker Registry configuration: – If pulling an image doesn't assume logging on to the corresponding server, specify the repository service URL. E.g. registry.hub.docker.com corresponds to public repositories on Docker Hub . – Otherwise, specify the corresponding Docker Registry configuration. Click New to create such a configuration.
New	Click this button to create a Docker Registry configuration. The Docker Registry dialog will open. (A Docker Registry configuration in PhpStorm represents your Docker image repository user account.)
Repository	The name of the repository (image) to be pulled, e.g. `centos` , `jboss/wildfly` .
Tag	The tag of the image to be pulled.

Push Image dialog

For this dialog to be available, the Docker integration plugin must be installed.

Specify the settings for pushing an image to a Docker image repository such as Docker Hub or Quay .

Item	Description
Registry	Select the Docker Registry configuration to be used. Click New to create such a configuration.
New	Click this button to create a Docker Registry configuration. The Docker Registry dialog will open. (A Docker Registry configuration in PhpStorm represents your Docker image repository user account.)
Repository	The name of the repository (image) you are pushing.
Tag	The tag for the repository (image) you are pushing.

Recent Changes Dialog

View | Recent Changes

`Shift+Alt+C`

In the Recent Changes popup, use the arrow keys to move in the list, and the `Enter` key to see details and, if necessary, to revert to the previous state.

Refactoring Dialogs

All refactoring dialogs provide similar controls to preview results and perform the refactoring.

Specific options and controls are described in the following topics:

- Copy Dialog
- Change Signature Dialog
- Change Signature Dialog for JavaScript
- Extract Constant Dialog
- Extract Field Dialog
- Extract Interface Dialog
- Extract Method Dialog
- Extract Variable Dialog
- Extract Variable Dialog for Sass
- Extract Parameter Dialog for JavaScript
- Inline Dialogs
- Make Static Dialogs
- Move Class Dialog
- Move File Dialog
- Move Static Member Dialog
- Move Namespace Dialog
- Pull Members up Dialog
- Push Members down Dialog
- Rename Dialogs
- Rename Dialog for a File
- Rename Dialog for a Table or Column
- Safe Delete Dialog

Copy Dialog

Refactor | Copy

F5

Use this dialog to specify the settings for the Copy refactoring.

Item	Description
New name	Specify the name of the file, or directory to be created.
To directory	Specify the directory where to create a copy. Select the destination directory from the list, or click ... (Shift+Enter) and select the directory in the Select target directory dialog that opens. (If necessary, you can create a new directory.)
Open copy in editor	Select this check box to automatically open a file, directory or package after it is copied. If you clear this check box, then the file, directory or package that you have copied will not be opened.

Change Signature Dialog

Refactor | Change Signature

`Ctrl+F6`

Use the Change Signature dialog to perform the Change Signature refactoring in the PHP context.

Use the available controls to make changes to the function signature. Specify how the function calls should be handled. Optionally, select the calling functions that the added parameters and exceptions (if any) should be propagated to.

Click Refactor to perform the refactoring right away. Click Preview to see the potential changes prior to actually performing the refactoring. (These will be shown in the Find tool window .)

Item	Description
Visibility	Select the method visibility scope (access level modifier) from the list.
Name	Use this field to modify the function name.
Parameters	See the description of the Parameters area .
Signature Preview	In this area, the current function signature is shown. (The information in this area is synchronized with the changes you are making to the function signature.)

Parameters area

Use the Parameters area to manage the function parameters.

The available controls let you add new parameters, remove the existing ones, reorder the parameters and also propagate new parameters to the calling functions (see the descriptions that follow).

In addition to that, you can change the names of the existing parameters.

To start editing a parameter, just click it. Alternatively, use the `Up` and `Down` arrow keys to move to the parameter of interest and `Enter` to start modifying it.

Item	Tooltip and shortcut	Description
+	Add `Alt+Insert`	Use this icon or shortcut to start adding a new parameter. Specify the name and default value in the corresponding fields. (The default parameter value is the value (or the expression) to be passed to the function in the function calls.) You can also propagate the parameters you have added to the calling functions.
−	Remove `Alt+Delete`	Use this icon or shortcut to delete the selected parameter.
↑	Up `Alt+Up`	Use this icon or shortcut to move the selected parameter one line up in the list of parameters.
↓	Down `Alt+Down`	Use this icon or shortcut to move the selected parameter one line down in the list of parameters.
	Propagate Parameters `Alt+G`	Use this icon or shortcut to propagate the added parameters to the calling functions. You can propagate the changes made to the function parameters to any function that directly or indirectly calls the function whose signature you are changing. (There may be the functions that call the current function. These, in their turn, may be called by other functions. You can propagate the changes to any of the methods in such sequences.) In the dialog that opens, select the functions you want the changes to be propagated to. Note that only the selected calling functions and the function calls within them will be affected. That is, the default values will be added into other function calls.
Create and initialize class properties		The check box is available only in the PHP context when the Change signature refactoring is invoked from the constructor of a class. − When this check box is selected and you add a parameter, this parameter is initialized as a field. PhpStorm creates a protected field with the same name as this parameter and adds a line with the following assignment: `$this-><parameter_name> = $<parameter_name>;` − When the check box is cleared, a parameter is added without initialization. For example, you have the following constructor:

```
class ChangeSignatureNewParam {
function __construct() {
$a = "Constructor in ChangeSignatureNewParam";
print $a;
}
}
```

If you invoke the Change signature refactoring from the __construct() method and add a new `$q` parameter, the result will depend on whether you select or clear the Create and initialize class properties check box:

The Create and initialize class properties check box is selected	The Create and initialize class properties check box is cleared

```
class ChangeSignatureNewParam {
private $q;
function __construct( $q ) {
$a = "Constructor in ChangeSignatureNewParam";
print $a;
$this->q = $q;
}
}
```

```
class ChangeSignatureNewParam {
function __construct( $q ) {
$a = "Constructor in ChangeSignatureNewParam";
print $a;
}
}
```

Change Signature Dialog for JavaScript

Refactor | Change Signature

`Ctrl+F6`

Use this dialog to change the function signature and to perform other, related tasks.

Item	Description
Name	Use this field to modify the function name.
Use the table and the controls to the right of it to manage the function parameters and their properties .	
Name	Use this field to specify the name of a parameter.
Value	A parameter value. If the parameter is a required one, the specified value (or expression) will be passed to the function in the function calls. If the parameter is an optional one, this value will be used in the function body to initialize the parameter.
Optional	Select this check box if the parameter is optional. Your selection will define how the parameter value is used.
or `Alt+Insert`	Use this icon or shortcut to start adding a new parameter. Specify the parameter name and value . Also, specify whether the parameter is optional . Note that you can propagate the parameters you have added to the calling methods.
or `Alt+Delete`	Use this icon or shortcut to delete the selected parameter.
or `Alt+Up`	Use this icon or shortcut to move the selected parameter one line up in the list of parameters.
or `Alt+Down`	Use this icon or shortcut to move the selected parameter one line down in the list of parameters.
or `Alt+G`	Use this icon or shortcut to propagate the added parameters to the calling methods. You can propagate new function parameters to any function that directly or indirectly calls the function whose signature you are changing. (There may be the functions that call the current function. These functions, in their turn, may be called by other functions. You can propagate new parameters to any of the functions in such sequences.) In the left-hand pane of the Select Methods to Propagate New Parameters dialog, expand the necessary nodes and select the check boxes next to the functions you want the new parameters to be propagated to.
Signature Preview	In this area, the current function signature is shown. (The information in this area is synchronized with the changes you are making to the function signature.)
Refactor	Click this button to perform the refactoring right away.
Preview	Click this button to see the expected changes prior to actually performing the refactoring.

Extract Constant Dialog

Refactor | Extract Constant

`Ctrl+Alt+C`

Item	Description
Name	In this text box, specify the name of the new constant.
Replace all occurrences	Select this check box to have PhpStorm automatically replace all the occurrences of the selected expression (if the selected expression is found more than once in the method).
Constant syntax	In this area, choose the method to define the PHP constant . The area is available only when the refactoring is invoked outside a class definition because the choice itself is relevant only in this context: extracted class constants are defined through the `const` keyword only. – define: choose this option to have PhpStorm define the constant through the define() function . – const: choose this option to have PhpStorm define the constant through the `const` keyword. This method preserves the approach accepted in PHP version 5.3.0. Note that such constants are defined during compilation therefore they should be declared at the top-level scope but not inside functions, loops or if statements.

Extract Field Dialog

Refactor | Introduce Field

`Ctrl+Alt+F`

Item	Description
Name	In this text box, specify the name of the new field.
Initialize in	In this area select where the new field will be initialized in.
Visibility	In this area, specify the visibility scope for the new field. The available options are: − Public - if you select this option, the new field will be accessible from anywhere. − Private - if you select this option, the new field will be accessible only from the current class. − Protected - if you select this option, the new field will be accessible from the current class as well as from its inherited and parent classes.
Replace all occurrences	Select this check box to have PhpStorm automatically replace all the occurrences of the selected expression. The check box is enabled only if the selected expression is used more than once in the class. All the found occurrences of the expression are highlighted.

Extract Interface Dialog

Refactor | Extract Interface

Item	Description
Interface Name	In this text box, specify the name of the interface to be created. By default, PhpStorm suggests the name of the current class from which the interface will be extracted. Update the name otherwise PhpStorm will report an error and the refactoring will no start.
Replace class reference with interface where possible	– When this check box is cleared, PhpStorm just creates an interface in a separate file and marks the source class as implementation of the new interface. – When the check box is selected, PhpStorm creates an interface and attempts to replace references to the source class with references to the interface in PhpDoc comments and parameters of methods. The proposed changes are compulsory displayed in a Refactoring Preview pane of the Find tool window. To have the interface extracted and the proposed changes applied, click Do Refactor.
Namespace	In this drop-dow list, specify the namespace the new interface will belong to. By default, the namespace of the source class in selected.
Target destination directory	In this drop-down list, specify the folder to store the dedicated file created for the new interface. By default, the field is read-only and shows the folder that corresponds to the namespace to which the interface will belong according to the PSR0 standard . To have the class with the interface stored in another folder, click F2 and specify the folder of your choice.
Members to form interface	This table shows all the methods detected in the current class. To have a method moved to the interface specified in the Move Member To Class, select the check box next to the method.
PhpDoc	In this area, specify how you want PhpDoc comments treated. The available options are: – As is: select this option to have the documentation left where it is in the source class. – Copy: select this option to have the inline documentation copied to the extracted interface without removing it from its current location. – Move: select this option to have the inline documentation moved to the extracted interface and deleted from its current location in the source class.

Extract Method Dialog

Refactor | Extract Method

`Ctrl+Alt+M`

The title of the dialog box may change to Extract Function:

- In the JavaScript context.
- In the PHP context, when the code selection is made inside a function or a script.

Item	Description
Name	In this text box, specify the name of the function or method to be generated on the basis of the selected source code.
Visibility	In this area, specify the visibility scope of the method to be generated. The available options are: – Public – Protected – Private This area is available only in the Extract Method dialog box when refactoring is invoked from a method of a PHP class.
Declare static	Select this check box to have a static method created. 1. If the new method cannot be declared static, or, vice-versa, can be created only as a static method, the Declare Static check box is disabled. 2. This check box is available only in the Extract Method dialog box when refactoring is invoked from a method of a PHP class.
Declare functional expression	Select this check box to have the new function defined through a function expression, for example, `new_method = function()`. The check box is only available when refactoring is invoked in the JavaScript context.
Output variable(s)	This read-only text box displays the name of the variable through which the output of the new method/function will be passed to the calling method/function. Depending on your choice in the Return output variable(s) through area, this variable either will be used in a return statement or will be declared as the passed by reference parameter of the new method/function.
Return output variable(s) through	In this area, specify the way in which the new method or function will return the output variables to the callee.. – Return statement - select this option to have the output variables returned by value. If the Output variable(s) read-only field shows exactly one output variable, it will be used as the return value. If the selection outputs several variables, these variables will be returned as an array. – Parameter(s) passed by reference - select this option to have the output variables returned by reference. PhpStorm will generate a method/function without a return statement. Instead, the output variables will be added to the set of input parameters in the method/function declaration. The names of these variables will be prepended with an ampersand `&`. The area is available only when refactoring is invoked in the PHP context.
Parameters	In this area, specify the parameters to be passed to the new method/function. If necessary, rename the parameters by double-clicking the corresponding parameter lines and entering new names. **Note** 1. If a parameter that is critical for the functionality of the new method is not selected, PhpStorm will be unable to proceed with the refactoring. 2. In the PHP context, the list may also contain output variables that will be passed by reference from the calling function. The names of these variables are prepended with an ampersand `&`.
Move Up/Down	Use these buttons to change the order of the parameters.
Replace tail "break/continue" statements with return statements	Select this check box to have PhpStorm transform tail break or continue statements if the selection contains any.
Signature preview	In this read-only field, view the declaration of the new method.

Extract Variable Dialog

Refactor | Extract Variable

`Ctrl+Alt+V`

Use this dialog box to extract:

- Variables in PHP or JavaScript contexts.
- Local variables for JavaScript 1.7 and higher.
- JavaScript constants.

Item	Description	Language
Name	Specify the name for the new variable.	All

Extract Variable Dialog for Sass

Refactor | Extract Variable

`Ctrl+Alt+V`

Use this dialog box to replace a Sass expression with a variable.

Item	Description
Name	Specify the name for the new variable.
Place for declaration	Select the place in the source code, where the new variable will be declared. The declaration can be global (a variable is declared in the beginning of a Sass file and is available throughout the whole file), or local (a variable is declared immediately before use, and is available in the current block only).
Replace all occurrences	Select this option to replace all the occurrences of the selected expression (if more than one occurrence of the selected expression is found).

Extract Parameter Dialog for JavaScript

Refactor | Extract | Parameter

`Ctrl+Alt+P`

Use this dialog to specify the options and settings related to the Extract Parameter refactoring in JavaScript .

Item	Description
Type	Specify the type of the new parameter. Usually, you don't need to change the type suggested by PhpStorm.
Name	Specify the name for the new parameter.
Value	The expression to be replaced with the new parameter. Initially, this field contains the expression that you have selected. Normally, this initial value does not need to be changed.
Optional parameter	If this option is selected, the new parameter is assigned a value in the function body. The value corresponds to that currently set in the Value field. The calls to the function do not change. If this option is not selected, all the function calls change according to the new function signature. The value specified in the Value field is added to the function calls and thus is passed to the function. No explicit value assignment for the new parameter is added to the function body.
Replace all occurrences	Select this option to replace all the occurrences of the selected expression within the function.

Inline Dialogs

Refactor | Inline

<context menu of a selection> | Refactor | Inline

Ctrl+Alt+N

Inline Variable dialog

The Inline Variable refactoring allows you to replace a redundant variable with its value. See examples .

To access the Inline Variable dialog box through a menu item or the keyboard shortcut, position the cursor at the variable to be inlined. The dialog box does not contain any controls but just displays the following message:

```
Inline variable <variable name>? (<the number of variable occurrences>)
```

Make Static Dialogs

Refactor | Make Static

Use this refactoring to convert an instance method into a static one.

The dialog box opens if you currently use the existing object via `this` or access the properties of the class. If your method does not need any parameters, nor rely on accessing properties or methods of this object, you will get the static method made silently, without displaying any dialog boxes:

Before After

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    static private function getFormattedDate() {
                    $format = getSettings()[ 'dateFormat' ];
                    return time( $format );
                    }
                    }
```

Make Method Static dialog

ItemDescription

Add object as a parameter with name

Select this check box to pass in an instance of the object via a parameter. In the text box below, specify the name of the parameter to be generated. After the refactoring, the new parameter will be documented in the PHPDoc block.

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = $this->getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    /**
                        * @param MyClass $instance
                        * @return
                        */
                    static private function getFormattedDate($instance) {
                    $format = $instance->getSettings()['dateFormat'];
                    return time($format);
                    }
                    }
```

Add parameters for fields

Use this area to pass the value of the property as a parameter instead of accessing the object inside the newly created static method if you access the properties of the class.

1. Select the Add parameters for fields check box.
2. In the Parameters list, which shows all the possibly suggested parameters, select the check boxes next to the ones that you want to pass the values in.

```
class MyClass {
                    private function getFormattedDate()
                    {
                    $format = $this->timeFormat;
                    return time($format);
                    }
                    }
```

```
class MyClass {
                    /**
                        * @param $timeFormat
                        * @return int;
                        */
                    static private function getFormattedDate($timeFormat) {
                    $format = timeFormat;
                    return time($format);
                    }
                    }
```

Move Class Dialog

Refactor | Move

F6

The dialog box opens when you position the cursor at the name of the class in the editor and invoke the refactoring.

PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when moving classes among namespaces.

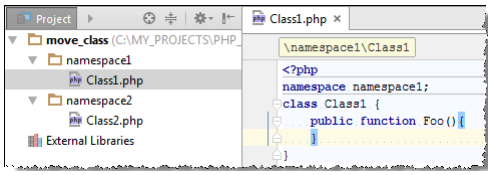
When you specify a namespace to move a class to, PhpStorm automatically updates the Target Destination Directory field, which shows the path to the folder where the file that implements the class in question will be stored.

Item	Description
Move class <class name> to namespace	When the dialog box opens, this field shows the namespace to which the class belongs. Specify the fully qualified name of the target namespace. Depending on your use case, this can be: – A new namespace under the current namespace in the format `<current namespace>\<new namespace>`. – Another subordinate namespace under the current parent in the format `<path to immediate parent namespace>\<namespace to move the class to>`. – An existing or a new namespace under another parent namespace in the format `<path to another root namespace>\<namespace to move the class to>`. Use only backslashes (`\`) as namespace separators.
Target destination directory	When the dialog box opens, the field shows the path to the folder where the file that implements the class to move is currently stored. The path is displayed in the format `...\<project root folder>\<path to the current namespace folder relative to the project root>`. The path is updated automatically as you specify the namespace to move the class to. However, if you are going to move a class to a non-existing namespace under another parent namespace, PhpStorm will not suggest the proper folder unless you appoint a root folder for your namespace structure by marking the relevant folder as Sources on the Directories page of the Settings dialog box, see Directories for details. Do one of the following: – Accept the preselected path displayed in the field. – Choose another path from the list. All of them are evaluated from the namespace root or from the current directory, so it is safe to choose any of them. – Click ... and select a folder in the dialog box that opens. – Press F2 and edit the preselected path. Keep in mind that this may cause problems with automatic loading in the future.
Search in comments and strings	If this check box is selected, PhpStorm will look for references to the class to be moved within comments and string literals in source code files.
Search for text occurrences	If this check box is selected, PhpStorm will look for references to the class to be moved within documentation, HTML, JSP, various configuration files (for example `.yml`), and other files included in your project.

Examples of Moving PHP Classes

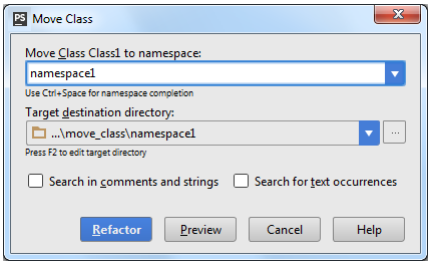
Moving a Class to a New Namespace under the Current Namespace

Suppose, your project `move_class` has a namespace `namespace1` with one class `Class1` implemented in the file `Class1.php`.

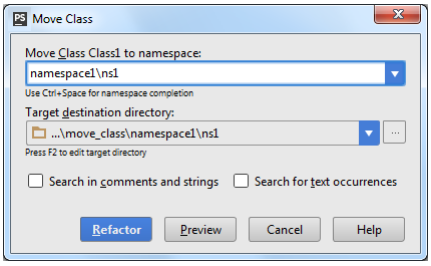


Let's create a subordinate namespace `namespace1\ns1` and move `Class1` to it.

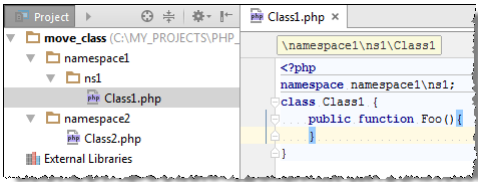
1. Open `Class1.php` in the editor, position the cursor at `Class1`, and press F6. The Move Class dialog box opens.



2. The Move class Class1 to namespace field shows the current namespace `namespace1`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace1` to `...\move_class\namespace1\ns1`.

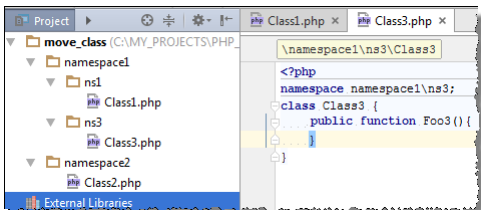


4. Press Refactor. As a result, PhpStorm creates a new subfolder `ns1` and moves the file `Class1` to it. Accordingly, the class `Class1` is moved to the `namespace1\ns1` namespace:



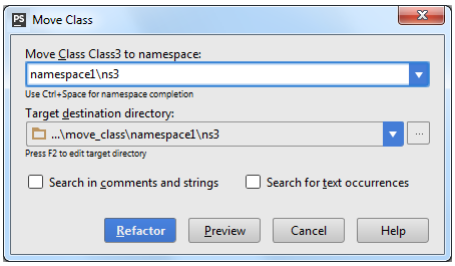
Moving a Class between Two Namespaces under the Same Parent

Suppose, your project `move_class` has a namespace `namespace1` with two subordinate namespaces `namespace1\ns1` and `namespace1\ns3`. These subordinate namespaces have `Class1` `Class3` respectively. In accordance with the PSR0 standard, these classes are implemented in the files `Class1.php` and `Class3.php` that are stored in the folders `...\move_class\namespace1\ns1` and `...\move_class\namespace1\ns3` respectively.

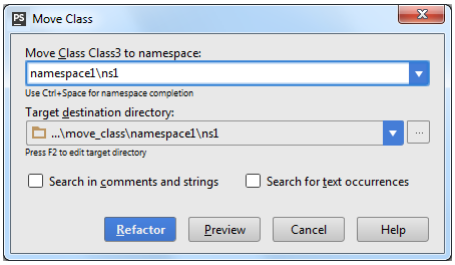


Let's move `Class3` from `namespace1\ns3` to `namespace1\ns1`.

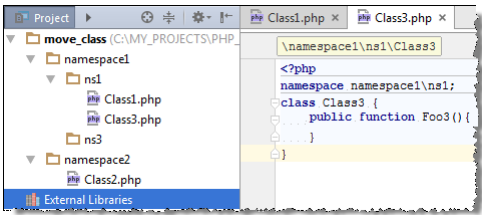
1. Open `Class3.php` in the editor, position the cursor at `Class3`, and press F6. The Move Class dialog box opens.



2. The Move class Class3 to namespace field shows the current namespace `namespace1\ns3`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace1\ns3` to `...\move_class\namespace1\ns1`.

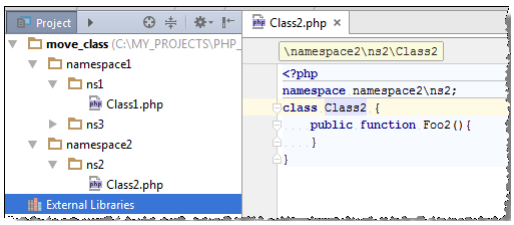


4. Press Refactor. As a result, PhpStorm moves the class `Class3` from `namespace1\ns3` to `namespace1\ns1`. Accordingly, the file `Class3` is moved from folder `...\move_class\namespace1\ns3` to folder `...\move_class\namespace1\ns1`:



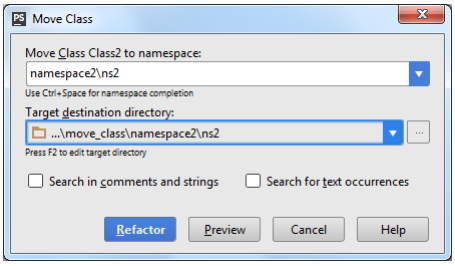
Moving a Class to an Existing Namespace under Another Parent Namespace

Suppose, your project `move_class` has two parent namespaces `namespace1` and `namespace2` with subordinate namespaces `namespace1\ns1` and `namespace2\ns2`. `Class2` is defined in `namespace2\ns2`, and in accordance with the PSR0 standard the file `Class2.php` that implements it is stored in `...\move_class\namespace2\ns2`.

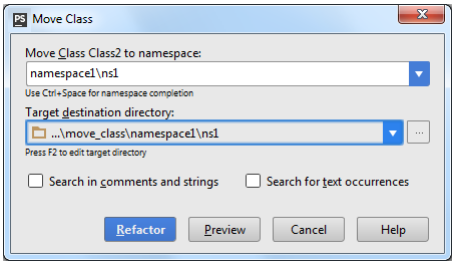


Let's move `Class2` from `namespace2\ns2` to `namespace1\ns1`.

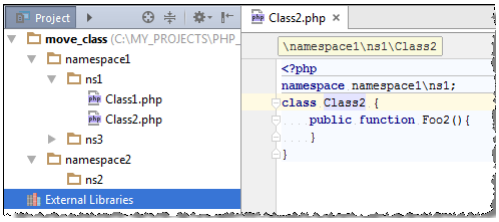
1. Open `Class2.php` in the editor, position the cursor at `Class2`, and press F6. The Move Class dialog box opens.



2. The Move class Class2 to namespace field shows the current namespace `namespace2\ns2`. Change it to `namespace1\ns1`.
3. As you type, the Target destination directory read-only field is updated automatically from `...\move_class\namespace2\ns2` to `...\move_class\namespace1\ns1`.

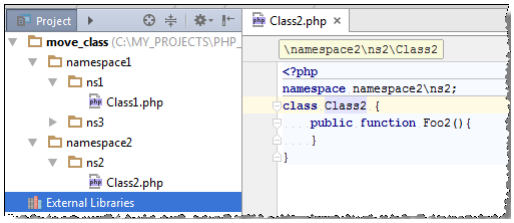


4. Press Refactor. As a result, PhpStorm moves the class `Class2` from `namespace2\ns2` to `namespace1\ns1`. Accordingly, the file `Class2` is moved from folder `...\move_class\namespace2\ns2` to folder `...\move_class\namespace1\ns1`:



Moving a Class to a New Namespace Outside the Immediate Parent Namespace

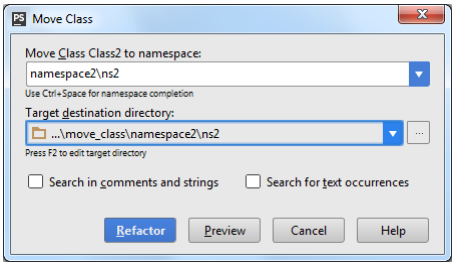
Suppose, your project `move_class` has two parent namespaces `namespace1` and `namespace2` with subordinate namespaces `namespace1\ns1` and `namespace2\ns2`. `Class2` is defined in `namespace2\ns2`, and in accordance with the PSR0 standard the file `Class2.php` that implements it is stored in `...\move_class\namespace2\ns2`.



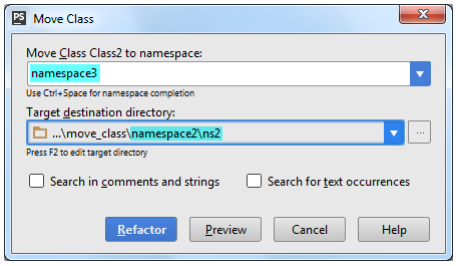
Let's move `Class2` from `namespace2\ns2` to a new namespace `namespace3` that does not exist yet. To meet the PSR0 requirements, a new folder `...\move_class\namespace3` for the `namespace3` namespace should be also created.

When starting the refactoring we should keep in mind that the new namespace is to be created outside the immediate parent namespace of our class's namespace, therefore PhpStorm will not suggest a proper folder for it. So we can either specify the target folder manually or appoint a root folder for namespaces which will be the starting point for PhpStorm to calculate the folder for the new namespace.

1. Open `Class2.php` in the editor, position the cursor at `Class2`, and press F6. The Move Class dialog box opens.



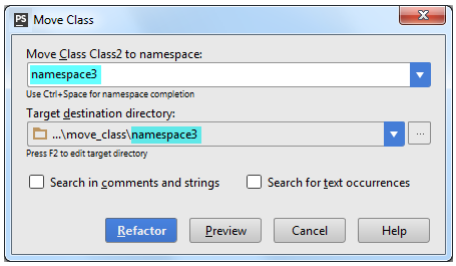
2. The Move class Class2 to namespace field shows the current namespace `namespace2\ns2`. Change it to `namespace3`. The result depends on whether you have appointed the root folder for your namespaces or not.
   - If no root folder is specified, the Target destination directory field still shows the path to `...\move_class\namespace2\ns2` because PhpStorm cannot calculate the path to the folder for the new namespace `namespace3` to be created:



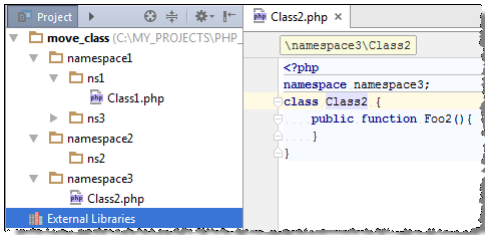
To solve the problem, press F2 and type the path manually:

```
...\move_class\namespace2\ns2
```

- Alternatively, specify the root folder for your namespaces by marking the `...\move_class\` folder as Sources as described in Configuring Folders Within a Content Root . Note, that this should be done before you start the refactoring, so if you have already opened the Move Class dialog box, close it first.
  After you have appointed the `...\move_class\` folder as the root for your namespaces, PhpStorm will calculate the folder for the new namespace and update the Target Destination Directory field automatically:



- Press Refactor. As a result, PhpStorm moves the class `Class2` from `namespace2\ns2` to `namespace3`. Accordingly, the file `Class2` is moved from folder `...\move_class\namespace2\ns2` to folder `...\move_class\namespace3`:



Move File Dialog

Refactor | Move

F6

The Refactor | Move menu command invokes one of the following dialog boxes depending on the selected item to be moved:

- The Move File dialog box is invoked when a file is selected in the Project tool window or opened in the editor.
- The Move File dialog box is invoked when a file opened in the editor.

Item	Description
To package	Specify the destination package. Type the path manually or click ... and select the target package in the dialog that opens .
Search in comments and strings	Select the check box to find and update the comments and strings to the file being moved.
Open moved in editor	Select this check box to see the moved file in the editor.

Move Static Member Dialog

Refactor | Move

F6

The Move Static Member refactoring dialog box is invoked from a PHP class for a static method, constant, or variable selected in the Structure view or in the editor.

Item	Description
Move member to class	In this text box, type the fully qualified name of the class where you want to move the members selected in the Members to be moved to another class list. PhpStorm provides completion for the class and namespace reference: as you type a class or namespace name, a suggestion list with the relevant item is displayed. You can also get a full list of available classes by pressing Ctrl+Space.
Members to be moved to another class	This table shows all the static methods, variables, and constants detected in the current class. To have a member moved to the class specified in the Move Member To Class, select the check box next to the class member.

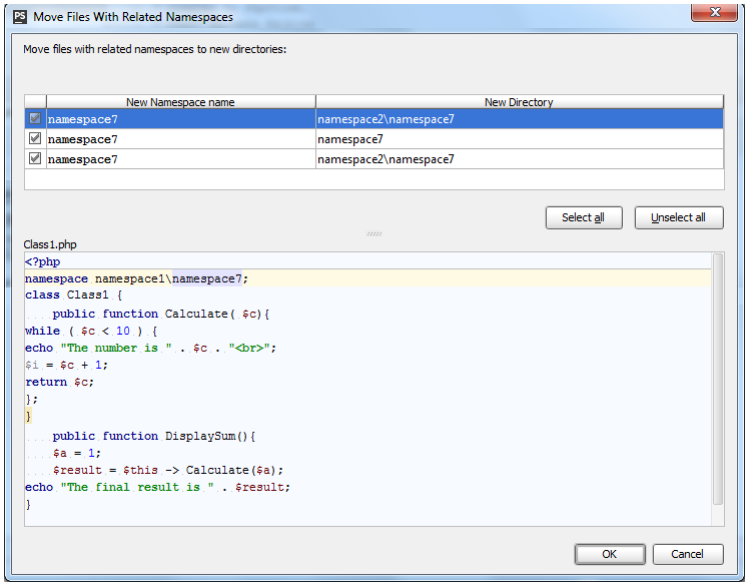
Move Namespace Dialog

The dialog box opens when you select a PHP namespace to be moved and choose Refactor | Move on the main menu or on the context menu of the selection.

PhpStorm supposes that the namespaces in your project are arranged in compliance with the PSR0 standard and enforces you to retain the folder structure and the namespace hierarchy in accordance with this standard when moving namespaces.

When you specify a namespace to move a namespace to, PhpStorm automatically updates the Target Destination Directory field, which shows the path to the folder that corresponds to the namespace in question.

Item	Description
New Namespace Name	When the dialog box opens, the field shows the fully qualified name of the selected namespace. Specify the new namespace name. Use only backslashes (`\`) as namespace separators.
Target Destination Directory	When the dialog box opens, the field shows the path to the folder which corresponds to the current namespace. The path is displayed in the format `...\<project root folder>\<path to the current namespace folder relative to the project root>`. The path is updated automatically as you specify the new namespace name. However, if you are going to move a namespace to another parent namespace, PhpStorm will not suggest the proper folder unless you appoint a root folder for your namespace structure by marking the relevant folder as Sources on the Directories page of the Settings dialog box, see Directories for details. Do one of the following: – Accept the preselected path displayed in the field. – Choose another path from the list. All of them are evaluated from the namespace root or from the current directory, so it is safe to choose any of them. – Click ... and select a folder in the dialog box that opens. – Press F2 and edit the preselected path. Keep in mind that this may cause problems with automatic loading in the future.
Refactor	Click this button to open the Move Files with Related Namespaces dialog box and specify the classes and files to be moved to the new namespace and the new folder. The upper pane of the dialog box lists the destination namespaces and folders for classes and files related to the namespace in question. Each item in the list corresponds to a class/file. When you move the cursor to an item, the bottom pane shows the contents of the file related with it.



– To have a class and the corresponding file moved to the destination namespace and the destination folder, select the check box next to the namespace/folder.
– To add all the items to the list or remove all of them from the list, click Select All or Unselect All respectively.

Pull Members Up Dialog

Refactor | Pull Members Up

Use this dialog to pull selected members up to the selected superclass.

Item	Description
Pull Up Members of <class_name> to	Select the destination object .
Members to be pulled up	In this area, select the members you want to move.

Push Members Down Dialog

Refactor | Push Members Down

Use this dialog to push selected members down to immediate subclass.

Rename Dialogs

Refactor | Rename

Shift+F6

Item	Description
Rename <symbol name> and its usages to	In this field, specify a new name for the symbol.
Search for references	This check box is only available for Rename File and Rename Directory refactoring. − Clear the check box to have PhpStorm rename only the file or folder itself without attempting to find its usages and update them accordingly. − Select this check box to have PhpStorm search for usages of the file/folder name and apply the refactoring to them.
Search in comments and strings	Select this check box to have the changes applied to comments and strings.
Search for text occurrences	Select this check box to have the changes applied to the documentation, HTML, and other files included in the project.

Rename Dialog for a File

Refactor | Rename

Shift+F6

Use this dialog to rename a file.

In addition to renaming the file itself, PhpStorm can also look for the usages of the file name. If found, the changes you are making to the file name can also be applied to these usages.

The usages are assigned to different categories which correspond to the options which you can turn on and off.

Note that regardless of the options selected, the search scope (that is, the places where the name occurrences are looked for) is always limited to the current entity (file, class, etc) and the entities that the current one depends on.

Item	Description
Rename <file> and its usages to	Specify a new name for the file.
Search for references	If this option is on, PhpStorm will look for occurrences of the file name in file references in source code files. Whether or not a string literal is a reference is defined by the (programming) language used in a source file.
Search in comments and strings	If this check box is selected, PhpStorm will look for occurrences of the file name within comments and string literals in your source code files.

Rename Dialog for a Table or Column

Refactor | Rename

Shift+F6

Use this dialog to rename a table or column.

In addition to renaming the table or column itself, PhpStorm can also look for the usages of the table or column name. If found, the changes you are making to the table or column name can also be applied to these usages.

Item	Description
Rename <table or column> and its usages to	Specify a new name for the table or column.
Search in comments and strings	If this check box is selected, PhpStorm will look for occurrences of the table or column name within comments and string literals in your source code files.
SQL Script	The statement to be run to rename the table or column. If necessary, you can edit the statement right in this pane.
Refactor	Execute the statement and make associated changes right away.
Preview	Preview potential associated changes prior to executing the statement. See Previewing changes .

Safe Delete Dialog

Refactor | Safe Delete

`Alt+Delete`

Use this dialog to define the scope of the Safe Delete refactoring.

Item	Description
Search in comments and strings	Select this check box to display the usages of the specified symbol in comments and strings in the Refactoring Preview tool window.
Search for text occurrences	Select this check box to apply the changes text files within the project (such as documentation, HTML, JSP, etc.)

Reformat File Dialog

Code | Reformat Code

`Ctrl+Alt+L`

The `Ctrl+Alt+L` command will try to reformat the source code of the specified scope automatically. This dialog box appears when you press `Ctrl+Shift+Alt+L` in the editor of the current file.

Item	Description
Only VCS changed text	If this check box is selected, then reformatting will apply only to the files that have been changed locally , but not yet checked in to the repository. This check box is only available for the files under version control.
Selected text	Choose this option to have the currently selected fragment of source code reformatted.
Whole file	Choose this option to have all the source code in the current file reformatted.
Optimize imports	Select this check box to remove unused import statements from the code within the selected scope.
Rearrange code	Select this check box to reorder your source code entries according to the configurations specified in the Arrangement tab of your Code Style settings . This check box is not available for Python files.
Run	Click this button to start reformatting the source code within the specified scope.

Reformat Code on Directory Dialog

Code | Reformat Code

`Ctrl+Alt+L`

This dialog box appears when you select a Reformat Code action on the directory.

Item	Description
Directory	This is a read-only field that displays the name of the selected directory.
Options	This area displays options for your code reformatting. You can select from the following options: – Include subdirectories: select this check box to have source code from files in the nested directories reformatted. This check box only appears when the nested directories exist. – Optimize imports: select this check box to remove unused import statements from the code within the selected scope. – Rearrange entries: select this check box to reorder your source code entries. – Only VCS changed text: if this check box is selected, then reformatting will apply only to the files that have been changed locally , but not yet checked in to the repository. This check box is only available for the files under version control.
Filters	This area displays options for the filters that you can apply to the code reformatting action. You can select from the following options: – Scope: select this check box to choose, from the drop-down list, a scope to which you want to apply your reformatting options. If you want to configure a custom scope, click ... button to open a Scopes dialog. – File mask(s): select this check box to choose, from the drop-down list, file extensions to which you want to apply your reformatting options.
Run	Click this button to start reformatting the source code within the specified scope.

Register New File Type Association Dialog

The dialog box opens when you attempt to open a file in the editor, but PhpStorm does not recognize its type by the file extension. Use the dialog box to associate unknown file extensions with existing file types.

Item	Description
Open matching files in PhpStorm	When this option is selected, PhpStorm treats the type of the file to be opened as one of the recognized file types. Choose the relevant type from the list box below, that displays all the file types recognized by PhpStorm.
File Pattern	In this text box, specify the file pattern to be associated with the selected file type. By default, the text box shows the following pattern: `*.<current file full extension>` .
Open matching files in associated application	When this option is selected, PhpStorm attempts to open the selected file using its native application, if this application is available.

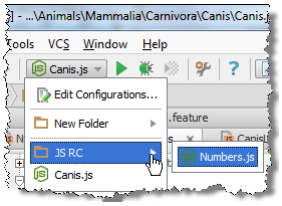
Tip You can change the association later in the File Types settings page.

Run/Debug Configurations

Run | Edit Configurations

Shift+Alt+F10

Shift+Alt+F9



Use this dialog box to create, edit, or remove run/debug configurations , and configure the default settings that will apply to all newly created run/debug configurations.

The default settings are grouped under the Defaults node in the left-hand part of the dialog box.

Click here for the description of the options that are common for all run/debug configurations.

Find the descriptions of specific run/debug configurations in the following pages:

- Run/Debug Configuration: App Engine for PHP
- Run/Debug Configuration:Behat
- Run/Debug Configuration: Chromium Remote
- Run/Debug Configuration: Compound
- Run/Debug Configuration: Cucumber.Js
- Run/Debug Configuration: Dart Command Line Application
- Run/Debug Configuration: Dart Remote Debug
- Run/Debug Configuration: Dart Test
- Run/Debug Configuration: Docker Deployment
- Run/Debug Configuration: Firefox Remote
- Run/Debug Configuration: Grunt.Js
- Run/Debug Configuration: Gulp.Js
- Run/Debug Configuration: JavaScript Debug
- Run/Debug Configuration: JSTestDriver
- Run/Debug Configuration: Karma
- Run/Debug Configuration: Meteor
- Run/Debug Configuration: Mocha
- Run/Debug Configuration: Node JS
- Run/Debug Configuration: Node JS Remote Debug
- Run/Debug Configuration: NodeUnit
- Run/Debug Configuration: Node-Webkit
- Run/Debug Configuration: NPM
- Run/Debug Configuration: PhoneGap/Cordova
- Run/Debug Configuration: PHP Built-in Web Server
- Run/Debug Configuration: PHP HTTP Request
- Run/Debug Configuration: PHP Remote Debug
- Run/Debug Configuration: PHP Script
- Run/Debug Configuration: PHPUnit
- Run/Debug Configuration: PHPUnit by HTTP
- Run/Debug Configuration: PHP Web Application
- Run/Debug Configuration: Spy-Js
- Run/Debug Configuration: Spy-Js for Node.js
- Run/Debug Configuration: XSLT

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of a new run/debug configuration. This field is not available for the default run/debug configurations .
Defaults	This node in the left-hand pane contains the default run/debug configuration settings. Select configuration to modify its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Before launch	Use the controls in this area to specify which tasks must performed before applying the run/debug configuration. The tasks are performed in the order they appear in the list. The following options are available: – + (`Alt+Insert`): click this icon to add a task to the list. Select one of the following task types: – Run External tool: select this option to run an application that is external to PhpStorm. In the dialog that opens , select the application(s) that you want to run. If the required application is not defined in PhpStorm yet, add its definition (for details see Configuring Third-Party Tools and External Tools). – Run Another Configuration: select this option to execute one on the existing run/debug configuration. Choose a configuration to execute from the dialog box that opens. – Compile TypeScript: select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Run File Watcher: select this option to run the File Watchers that are active in the current project. For more information, see Using File Watchers . – Generate CoffeeScript Source Maps: select this option to generate source maps for your CoffeeScript sources. In the dialog that opens, specify the path to the folder where CoffeeScript files will be generated. For more information, see CoffeeScript Support . – Run Remote External tool: select this option to run a remote SSH external tool. In the dialog that opens, specify the SSH external tool that you want to run. For more information, see Remote SSH External Tools . – – (`Alt+Delete`): click this icon to remove the selected task from the list. – (`Enter`): click this icon to modify the selected task. Edit the task settings in the dialog that opens. – ↑ (`Alt+Up`): click this icon to move the selected task up in the list. – ↓ (`Alt+Down`): click this icon to move the selected task down in the list. – Show this page: select this check box if you want to display the run/debug configuration settings before applying it. – Active tool window: select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Item	Description
Confirm rerun with process termination	The behavior of this button depends on selecting the check box Single instance only for a particular run/debug configuration. – If this check box is selected, then, in case of a single instance, launching a new process (for example, by clicking ▶ on the main toolbar) while the other process is still running, results in showing a dialog box, where one should confirm termination of the current process and launching a new one. – If this check box is not selected (or in case of multiple instances), PhpStorm starts the new process silently.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.

Run/Debug Configuration: App Engine for PHP

In this dialog box, create configurations to run and debug PHP applications locally on the PHP Development Server before deploying them to the Google PHP Runtime Environment .

The dialog box is available only if the Google App Engine Support for PHP plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

– Configuration settings specific for App Engine for PHP

– Command Line area

– Toolbar

– Common options

Configuration settings specific for App Engine for PHP

Item	Description
Name	In this text box, specify the name of the run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Host	In this field, specify the host to run the development server and the application on, the default is `localhost` .
Port	In this field, specify the port through which PhpStorm will communicate with the development server, the default port is `8080` .

Command Line area

In this area, specify the settings for running and debugging your application on the PHP development server in the command-line mode .

Item	Description
Interpreter options	In this field, specify the options to be passed to the PHP executable file of the built-in PHP interpreter, see Command-Line Arguments for details.
Yaml files	In this field, specify the `.yaml` configuration files to use. This field is optional, use it when your application consists of several modules and therefore several `.yaml` configuration files are used.
Custom working directory	In this text box, specify the location of the files that are outside the folder with your sources and are referenced through relative paths. Type the path manually or click the Browse button and select the desired folder in the dialog that opens .
Environment variables	In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list. **Item Keyboard Description**

	shortcut	
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details. – Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration:Behat

This dialog box is available only when the PHP and Behat plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Use this dialog box to create a configuration to be used for running and debugging unit tests on PHP applications using the Behat framework.

On this page:

- Before you start
- Test Runner area
- Command Line area
- Toolbar
- Common options

Before you start

To run Behat tests:

1. Install and configure the Behat framework on your computer as described in Using Behat Framework .
2. Make sure the PHP and Behat plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Test Runner area

In this area, specify the scenarios to launch and the command line switches to be passed to Behat.

Item	Description
Test scope	In this area, specify the location of scenarios or the configuration file where they are listed. – Directory: select this option to have all the scenarios in a directory launched. In the Directory text box, specify the directory to search for `.feature` files with scenarios in. Type the path to the directory manually or click the Browse button ... and select the desired directory in the Choose Test Directory dialog box, that opens. – File: select this option to have all the scenarios in a specific `.feature` file launched. 1. In the File text box, specify the `.feature` file to search the scenarios in. Type the path to the file manually or click the Browse button ... and select the desired directory in the dialog box, that opens. 2. In the Class text box, specify the desired class. Type the class name manually or click the Browse button ... and select the desired class in the tree view, that opens. – Scenario: select this option to have a specific scenario launched. 1. In the File text box, specify the `.feature` file to search for the scenario in. Type the file name manually or click the Browse button ... and select the desired file in the tree view, that opens. 2. In the Scenario text box, specify the desired scenario. – Defined in the configuration file: select this option to have Behat execute the tests from a dedicated `.yaml` configuration file. By default, Behat uses the configuration file appointed in the Test Runner area of the Behat . In its turn, this can be either the `behat.yaml` file, for which Behat searches in the project root or in the `config` folder, or any other `.yaml` configuration file which you specified as Default during the initial configuration of Behat in PhpStorm. – To have the default for all Behat run configurations file used, clear the Use alternative configuration file check box. – To launch scenarios from a custom configuration file, select the Use alternative configuration file check box and specify the location of the desired `.yaml` file in the text box next to it. – To open the Behat page and specify another default configuration file to use, click the button.

Command Line area

In this area, customize the behavior of the current PHP interpreter by specifying the options and arguments to be passed to the PHP executable file.

Item	Description
Interpreter options	In this text box, specify the options to be passed to the PHP executable file. They override the default behavior of the PHP interpreter and/or ensure that additional activities are performed. If necessary, click and type the desired options in the Command Line Options dialog box. Type each option on a new line. When you close the dialog box, they are all displayed in the Command line options text box with spaces as separators.
Custom working directory	In this text box, specify the location of the files that are outside the folder with tests and are referenced in your tests through relative paths. Type the path manually or click the Browse button ... and select the desired folder in the dialog that opens . This settings does not block the test execution because the location of tests is always specified through a full path to the corresponding files and/or directories.
Environment variables	In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
–	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Chromium Remote

Use this dialog box to create configurations for debugging applications remotely in ChromiumOS .

On this page:

- Chromium Remote-specific configuration settings
- Toolbar
- Common options

Chromium Remote-specific configuration settings

Item	Description
Host	In this text box, specify the host where the application is running.
Port	In this spin box, specify the port to connect to.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated.

In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .

- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Compound

Use this dialog box to create configurations containing multiple run/debug configurations that you can launch at once. This is useful, for example, if you want to launch various automated tests and get test results in one window.

Press + to select which of the existing configurations you want to include into the Compound configuration, and fill in the following fields:

Item	Description
Name	Specify the name of the new Run/Debug Configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.

Run/Debug Configuration: Cucumber.js

In this dialog box, create configurations for running and debugging JavaScript unit tests using the Cucumber.js test runner.

On this page:

- Getting access to the Run/Debug Configuration: Cucumber.js dialog
- Cucumber.js-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Cucumber.js dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Using the Node Package Manager, install the Cucumber.js as described in Preparing to Use Cucumber.Js Test Runner .
4. Install and enable the Cucumber.js and Gherkin plugins. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Cucumber.js-specific configuration settings

Item	Description
Feature file or directory	In this text box, specify the tests to run. Cucumber.js runs tests that are called features and are written in the Gherkin language. Each feature is described in a separate file with the extension `feature`. – To have one feature executed, specify the path to the corresponding `.feature` file. – To have a bunch of features executed, specify the folder where the `.feature` files to run are stored. Type the path manually or click the Browse button ... and choose the file or folder in the dialog box that opens.
Cucumber.js arguments	In this text box, specify the command line arguments to be passed to the Cucumber.js executable file, such as `-r` (`--require LIBRARY|DIR`), `-t` (`--tags TAG_EXPRESSION`), or `--coffee` . For details, see Cucumber's native built-in help available through the following command: `cucumber-js --help`
Executable path	In this text box, specify the location of the cucumber executable file, `.cmd` , `.bat` , or other depending on the operating system used. The location depends on the installation mode, see Installing the Cucumber.js test runner . Type the path manually or click the Browse button ... and choose the file in the dialog box that opens.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item Keyboard Description

	shortcut	
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Dart Command Line Application

In this dialog box, create a configuration to be used for running and debugging the client-side Dart code.

On this page:

- Getting access to the Run/Debug Configuration: Dart Command Line Application dialog
- Dart Command Line Application-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Dart Command Line Application dialog

1. Make sure the Dart development environment is configured, see Preparing for Dart Development .
2. Make sure the Node.js and Dart plugins are activated. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Dart Command Line Application-specific configuration settings

Item	Description
Name	In this text box, specify the name of the run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Dart file	In this text box, specify the Dart file to start the application from. The file must contain a `main()` method. Type the path to the file manually or click the Browse button ... and select the file in the dialog box that opens.
VM options	In this text box, specify the options to launch the Dart Virtual Machine with. When specifying the options, follow these rules: – Use spaces to separate individual options, for example, `-client -ea -Xmx1024m` . – If an option includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"` . – If an option includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value\"` .
Program arguments	In this text box, type the list of arguments to start the application with. Use the same format as for VM options.
Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.
Working Directory	In this field, specify the working directory of the application. All references in the starting Dart application file, for example, imports, will be resolved relative to this folder, unless such references use full paths. By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the

`.idea\workspace.xml` .

This check box is not available when editing the run/debug configuration defaults.

Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
↑	Alt+Up	Click this icon to move the selected task one line up in the list.
↓	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Dart Remote Debug

In this dialog box, create configurations to debug Dart applications on remote hosts.

On this page:

- Getting access to the Run/Debug Configuration: Dart Remote Debug dialog
- Dart Unit-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Dart Remote Debug dialog

1. Make sure the Dart development environment is configured, see Preparing for Dart Development .
2. Make sure the Node.js and Dart plugins are activated. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Dart Unit-specific configuration settings

Item	Description
Host	In this field, specify the host where the Dart application is running.
Port	In this field, specify the port for the debugger to communicate with the specified host. The specified port is shown in the Command line arguments for the remote Dart VM read-only field. Note that the application must be started exactly with these arguments, so do not forget to provide this information to the one who will start it on the remote host or specify the arguments in the Dart Remote Debug run configuration if you are going to launch the application yourself on the localhost.
Search Sources in	From this drop-down list, choose the Dart project to debug if your PhpStorm project contains several Dart projects configured as content roots. See Developing Several Dart Projects (Packages) within one PhpStorm Project on Preparing for Dart Development .
Command Line Arguments for the Remote Dart VM	This read-only field shows the port specified in the Port field. Note that the application must be started exactly with these arguments, so do not forget to provide this information to the one who will start it on the remote host or specify the arguments in the Dart Remote Debug run configuration if you are going to launch the application yourself on the localhost.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: − Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . − Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. − Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File

Watchers for details.

- Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Dart Test

In this dialog box, create configurations to run unit tests for Dart applications.

On this page:

- Getting access to the Run/Debug Configuration: Dart Test dialog
- Dart Test-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Dart Test dialog

1. Make sure the Dart development environment is configured, see Preparing for Dart Development .
2. Make sure the Node.js and Dart plugins are activated. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

Dart Test-specific configuration settings

Item	Description

Test Mode	From this drop-down list, choose the scope of tests to execute. The available options are:
	– All in File: choose this option to to have PhpStorm run all the test methods in a file and specify this file in the Test File text box.
	– All in Folder: choose this option to to have PhpStorm run all the test methods from the files stored in a folder and specify this folder in the Test Folder text box.
	– Group or test by name: choose this option to to have PhpStorm run a specific tast or a group of tests.
	  – In the Test File text box, specify the file where the test or the group is defined.
	  – In the Group or Test Name text box, type the name of the test or the group of tests to execute.
	
	See more about grouping tests at https://www.dartlang.org/articles/dart-unit-tests/#grouping-tests .

Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Docker Deployment

Run | Edit Configurations | + | Docker Deployment

Docker Deployment run/debug configurations let you download and build Docker images, and create and start Docker containers.

To make Docker Deployment run/debug configurations available, you must download and install the Docker integration plugin , see Installing, Updating and Uninstalling Repository Plugins .

- Name, Share, and Single instance only
- Deployment tab
- Container tab
- Before Launch options
- Toolbar

See also, Docker .

Name, Share, and Single instance only

Item	Description
Name	The name of the run configuration.
Share	Select this check box to share the run configuration through version control. If the check box is not selected, the run configuration settings are stored in `.idea/workspace.xml` . If the check box is selected, the settings are stored in a separate `.xml` file in `.idea/runConfigurations` or in the `.ipr` file.
Single instance only	If you select this check box, only one instance of the run configuration will run at a time.

Deployment tab

Item	Description
Server	Select the Docker configuration to be used. To create a new configuration, or to edit an existing one, click ... (`Shift+Enter`). For more information, see Docker.
Deployment	One of the following: - Docker Image. Select this option if you want to deploy a Docker image available locally. - <docker-dir>/Dockerfile. Select the Dockerfile to be used. (The Dockerfiles available in your project are suggested.) - <docker-dir>/docker-compose.yml. Select the Docker Compose YAML file to be used. (The docker-compose.yml files available in your project are suggested.)
The following settings are not available if a `docker-compose.yml` file is selected in the Deployment field.	
Image ID	If Docker Image is selected in the Deployment field: the ID of the image to be deployed, a sequence of hexadecimal symbols. For the image of interest, you can, for example, use the Copy image ID command in the Application Servers tool window and then paste the ID into this field.
Image tag	If <docker-dir>/Dockerfile is selected in the Deployment field: if you want to use an image with a particular tag, specify that tag. (The image in this case is specified in the Dockerfile.) If nothing is specified, Docker will look for the image with the tag `latest` .
Container name	If you want to give the container that will be created a particular name, specify that name. Otherwise, Docker will itself decide what the name should be.
After launch	Select this check box to start a web browser after starting the run/debug configuration. Select the browser from the list. Click ... (`Shift+Enter`) to configure your web browsers.
With JavaScript debugger	If this check box is selected, the web browser is started with the JavaScript debugger enabled. Note that JavaScript debugging is available only for Firefox and Google Chrome. When you debug your JavaScript in Firefox for the first time, the JetBrains Firefox extension is installed.
The field underneath After launch	Specify the URL the browser should go to when started. In most typical cases, this URL corresponds to the root of your Web application or its starting page.
Debug Port	The port to be used for debugging. (Debugging is supported only for Java.) This may be any unused port on your computer. If there is a warning *Debug port forwarding not found*, click Fix. As a result, the necessary port mapping info is added to the JSON container settings file . Note the command line arguments suggested by PhpStorm. To make debugging possible, these should be used to start the JVM in the container.

Container tab

This tab contains the settings for your container.

NOTE: The settings on this tab are ignored if a `docker-compose.yml` file is selected in the Deployment field .

Item	Description
JSON file	Path to a JSON file with container settings in the Docker Remote API format. - ... lets you select an existing file. - lets you generate and save a sample container settings `.json` file. CLI lets you translate the `docker run` options into JSON container settings Remote API format. The result is saved in a JSON file which is associated with your container. (For `docker run [OPTIONS] IMAGE [COMMAND] [ARG...]` `OPTIONS` are specified in the CLI options field of the dialog that opens when you click CLI, `COMMAND` is specified in the Command field,

	`ARG...` part is specified in the Entrypoint or Command filed.)
The settings below the JSON file field take precedence when overlapping ones set in the JSON file.	
Entrypoint	The entry point for your container, see e.g. ENTRYPOINT (default command to execute at runtime) in Docker run reference .
Command	See CMD (default command or options) in Docker run reference .
Publish all ports	If the check box is selected, all the container's exposed ports are bound to a random host port. See e.g. EXPOSE (incoming ports) in Docker run reference .
Port bindings	Container port/protocol - Host IP address/port mappings. E.g. `8080 tcp 127.0.0.1 18080` would bind the TCP port `8080` inside the container to port `18080` on the `localhost` or `127.0.0.1` interface on the host machine. For more info, see e.g. Connect using network port mapping in Linking containers together .
Links	Container name - alias mappings. See e.g. Communication across links in Linking containers together .
Volume bindings	Volume bindings such as container path - host path. See e.g. Managing data in containers .
Environment variables	Environment variables as variable - value pairs. See e.g. ENV (environment variables) in Docker run reference .

Before Launch options

Specify which tasks should be carried out before starting the run/debug configuration.

Item	Shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Build Artifacts. Select this option to build an artifact or artifacts. In the dialog that opens, select the artifact or artifacts that should be built. See also, Working with Artifacts . – Run Another Configuration. Select this option to execute another run/debug configuration. In the dialog that opens, select the configuration to be run. – Run Ant target. Select this option to run an Ant target. In the dialog that opens, select the target to be run. For more information, see Ant . – Generate CoffeeScript Source Maps. Select this option to generate the source maps for your CoffeeScript sources. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Maven Goal. Select this option to run a Maven goal. In the dialog that opens, select the goal to be run. For more information, see Maven . – Run Remote External Tool. Select this option to run a remote application which is external to PhpStorm. In the dialog that opens, select one or more remote applications to be run. If the necessary applications are not defined in PhpStorm yet, add their definitions. For more information, see Configuring Third-Party Tools and External Tools .
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
↑↓	Alt+Up / Alt+Down	Click these icons to move the selected task one line up or down in the list. (The tasks are performed in the order that they appear in the list.)
Show this page		Select this check box to show the run/debug configuration settings prior to actually starting the run/debug configuration.
Activate tool window		If this check box is selected, the Debug tool window opens when you start the run/debug configuration in the debug mode. Otherwise, the tool window isn't shown. However, when the configuration is running in the debug mode, you can open the Debug tool window for it yourself if necessary.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Create a run/debug configuration.
–	Alt+Delete	Delete the selected run/debug configuration.
	Ctrl+D	Create a copy of the selected run/debug configuration.
		View and edit the default settings for the selected run/debug configuration.
↑↓	Alt+Up / Alt+Down	Move the selected run/debug configuration up and down in the list. The order of configurations in the list defines the order in which the configurations appear in the corresponding list on the main toolbar.
		You can group run/debug configurations by placing them into folders. To create a folder, select the configurations to be grouped and click . Specify the name of the folder. Then, to move a configuration into a folder, between the folders or out of a folder, use ↑ and ↓. You can also drag a configuration into a folder. To remove grouping, select a folder and click –. See also, Creating Folders and Grouping Run/Debug Configurations .

Run/Debug Configuration: Firefox Remote

In this dialog box, create configurations to use for debugging applications locally in Firefox, versions 36 and higher. Note that remote debugging in Firefox is currently not supported at all.

On this page:

- Firefox Remote-specific configuration settings
- Toolbar
- Common options

Firefox Remote-specific configuration settings

Item	Description
Host	In this text box, specify the host where the application is running. Currently it is just `localhost`.
Port	In this spin box, specify the port the debugger will listen to. It must be the port that you specified when enabling debugging in Firefox, see Debugging JavaScript. The default value is 6000.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools. – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will

be launched.

- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Grunt.js

In this dialog box, create configurations for running Grunt.js tasks.

On this page:

- Getting access to the Run/Debug Configuration: Grunt dialog
- Grunt-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Grunt dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Install Grunt as described in Installing Grunt .

Grunt-specific configuration settings

Item	Description
Gruntfile	In this field, specify the location of the `Gruntfile.js` file to retrieve the definitions of the tasks from. Select the path from the drop-down list or click the ... button and choose the file from the dialog box that opens.
Tasks	In this field, specify the tasks to run. Do one of the following: – To run one task, select it from the drop-down list. – To run several tasks, type their names in the text box using blank spaces as separators.
Force execution	Select this check box to have Grunt ignore warnings and continue executing the launched task until the task is completed successfully or an error occurs, If the check box is cleared, the task execution is stopped by the first reported warning.
Verbose mode	Select this check box to have the verbose mode applied and thus have a full detailed log of a task execution displayed.
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
Node Options	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. See Node Parameters for details.
Grunt-cli Package	In this field, specify the path to the globally installed `Grunt-cli` package installed globally See Installing Grunt.js for details.
Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.

Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Gulp.js

In this dialog box, create configurations for running Gulp.js tasks.

On this page:

- Getting access to the Run/Debug Configuration: Gulp.js dialog
- Gulp.js-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Gulp.js dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Install the `Gulp` package as described in Installing Gulp.js .

Gulp.js-specific configuration settings

Item	Description
Gulpfile	In this field, specify the location of the `Gulpfile.js` file to retrieve the definitions of the tasks from. Select the path from the drop-down list or click the ... button and choose the file from the dialog box that opens.
Tasks	In this field, specify the tasks to run. Do one of the following: – To run one task, select it from the drop-down list. – To run several tasks, type their names in the text box using blank spaces as separators.
Arguments	In this text box, specify the arguments for tasks to be executed with. Use the following format: `--<parameter_name> <parameter_value>` For example: `--env development` . For details about passing task arguments, see https://github.com/gulpjs/gulp/blob/master/docs/recipes/pass-arguments-from-cli.md
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
Node Options	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. See Node Parameters for details. In the default configuration, type `--harmony` in this text box to have PhpStorm build a tasks tree according to a `Gulfile.js` written in ECMA6. Technically, PhpStorm invokes Gulp.js and processes `Gulpfile.js` according to the default Gulp.js run configuration. However this is done silently and does not require any steps from your side. However, if your `Gulpfile.js` is written in ECMA6 , by default PhpStorm does not recognize this format and fails to build a tasks tree. To solve this problem, specify `--harmony` as a Node parameter of the default Gulp.js run configuration.
Gulp Package	In this field, specify the path to the `Gulp` package installed locally, under the project root. See Installing Gulp.js for details.
Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.

Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

−	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: JavaScript Debug

In this dialog box, create a configuration to be used for debugging JavaScript sources locally and remotely and for debugging Dart Web applications.

1. Local debugging. In this mode, the application files remain where they are in the PhpStorm project on you machine. PhpStorm runs them on the built-in Web server with the root URL `http://localhost:<built-in server port>/<project root>`. The built-in server port (1024 or higher) is specified on the Data Views page of the Settings dialog box. A debugging session in this mode can be initiated in two ways:
   - Open the HTML file with the JavaScript injection to be tested and launch the temporary run/debug configuration that PhpStorm has generated automatically. This approach it helpful when you do not need to debug the entire application but just one script.
   - Create and launch a permanent debug configuration.
2. Remote debugging. In this mode, the application files are on an external web server and you have their copies in a PhpStorm project on your computer. No matter, whether the web server itself is running on a physically remote host or on your machine, application files deployed on it are treated as remote. For details, see Working with Web Servers: Copying Files .
   You can deploy the application files to a web server and launch the application yourself or debug an already running application.

   When a remote HTML file with a JavaScript injection is opened, the debugger tells PhpStorm the name of the currently processed file and the number of the line to be processed. PhpStorm opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called mapping, it is set in the debug configuration.

   A Remote debugging session can be initiated only through a permanent debug configuration that you have to define yourself manually.

PhpStorm supports local and remote JavaScript debugging only in Google Chrome and other browsers of the Chrome family. In Firefox , remote debugging is not supported at all. Local debugging is supported for Firefox, version 36 an higher, through the Firefox Remote debug configuration.

PhpStorm supports local and remote JavaScript debugging only in Google Chrome and other browsers of the Chrome family. In Firefox , remote debugging is not supported at all. Local debugging is supported for Firefox, version 36 an higher, through the Firefox Remote debug configuration.

On this page:

- JavaScript Debug-specific configuration settings
- Toolbar
- Common options

JavaScript Debug-specific configuration settings

Item	Description
Name	In this text box, specify the name of the current run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
URL	- Debugging JavaScript: In this text box, specify the URL address of the HTML file that references the JavaScript to debug. For local debugging, type the URL in the format `http://localhost:<built-in server port>/<project root>`. The built-in server port (1024 or higher) is specified on the Data Views page of the Settings dialog box. - Debugging a Dart Web application: In this text box, specify the URL address of the HTML file the references the Dart code to debug. Make sure that the port specified in this URL address is the same as the Built-in server port on the Debugger page of the Settings dialog box and the port specified in the settings for the Chrome Extension.
Browser	From this drop down list, select the browser, where your application will be debugged. - Debugging JavaScript: Chrome - Debugging a Dart Web application: If you choose Dartium which has a built-in Dart virtual machine, simply the Dart code is executed. If you choose Chrome, the Dart code is compiled into JavaScript through the Dart2js tool, which is a part of the Pub tool. Whether such compilation is required or not is decided by the Pub Serve tool. The tool is invoked automatically when you run or debug a Dart web application, either by opening an HTML file or by launching a run/debug configuration. The tool analyzes the `<script></script>` element in the HTML file. If the element references a Dart file, Pub Serve invokes the Dart2js compiler whereupon the compiled JavaScript files are passed to the browser where they are processed. Specify the URL address of the HTML file the references the Dart code to debug. Make sure that the port specified in this URL address is the same as the Built-in server port on the Debugger page of the Settings dialog box and the port specified in the settings for the Chrome Extension.
Remote URLs of local files	- Debugging JavaScript: PhpStorm displays this area only when you create a permanent debug configuration manually. For automatically generated temporary configurations the area is not shown. In this area, map the local files to be involved in debugging to the URL addresses of their copies on the server. - File/Directory - in this read-only field, select the desired local file or directory in the project tree. - Remote URL - in this text box, type the absolute URL address of the corresponding file or folder on the server. These mappings are required only if the local folder structure under the project root differs from the folder structure on the server. If the structures are identical, PhpStorm itself "retrieves" the paths to local files by parsing the URL addresses of their copies on the server. - Debugging a Dart Web application: PhpStorm displays this area only when the port specified in the URL field is different from the port of the built-in Web server specified on the Debugger page of the Settings dialog box.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−		Click this button to remove the selected configuration from the list.

	Alt+Delete	
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: JSTestDriver

In this dialog box, create a configuration to be used for running JavaScript unit tests in the browser against a JSTestDriver server. Configurations of this type enable running unit tests based on the JSTestDriver Assertion , Jasmine , and QUnit frameworks.

The dialog box is available when the JSTestDriver plugin is activated. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

On this page:

- Configuration tab
  - Test
  - Server
- Debug tab
- Coverage tab
- Toolbar
- Common options

Configuration tab

In this tab, specify the tests to run, the configuration files to use, and the server to run the tests against.

Test

In this area, tell PhpStorm where to find the tests to execute and how to get test runner configuration files that define which test files to load and in which order. The main approaches are:

- Specify the location of one or several previously created configuration files.
- Point at the target test file, test case, or test method, and then specify the location of the corresponding configuration file.

The way to find tests and configuration files is defined in the Test drop-down list. This choice determines the set of other controls in the area.

Item	Description	Available for
Test	In this drop-down list, specify how PhpStorm will get test runner configuration files. – All configuration files in directory: select this option to use all the test runner configuration files in a specific folder. – Configuration file: select this option to use a specific test runner configuration file. – JavaScript test file: select this option to have tests from a specific file executed using one of the previously defined configuration files. – Case: select this option to run a specific test case using one of the previously defined configuration files. – Method: select this option to run a specific test method using one of the previously defined configuration files.	
Directory	In this text box, specify the folder to look for test runner configuration files in. Type the path manually or click the Browse button ... and select the required folder in the dialog box, that opens.	All configuration files in directory
Matched configuration files	This read-only field shows a list of all the `*jstd` and `JSTestDriver.conf` test runner configuration files detected in the specified folder .	All configuration files in directory
Configuration file	In this text box, specify the test runner configuration file to use. Type the path manually or click the Browse button ... and select the required file in the dialog box that opens.	Configuration file
JS test file	In this text box, specify the JavaScript files with tests to be executed. Type the path manually or click the Browse button ... and select the required file in the dialog box, that opens.	JavaScript test file Case Method
Case	In this text box, type the name of the target case from the specified JavaScript file .	Case Method
Method	In this text box, type the name of the target method from the specified test case within the specified JavaScript file .	Method

Server

In this area, appoint the test server to run tests against.

Item	Description
At address	Choose this option to have the test execution handles by a remote test server. In the text box, specify the URL address to access the server through.
Running in IDE	Choose this option to have test execution handles through the JSTestDriver server that comes bundled with PhpStorm and can be launched from it.
Test Connection	Click this button to check that the specified test server is accessible. The server must be running. Start the server from PhpStorm or manually, according to the server-specific instructions.

Debug tab

In this tab, appoint the browser to debug the unit test in when two or more browsers are captured simultaneously.

Item	Description
Debug	From this drop-down list, choose the browser to debug the specified tests in when two browsers are captured at a time. The available options are: – Chrome – Firefox

Coverage tab

In this tab, specify the files that you do not want to be involved in coverage analysis.

ItemDescription	
Exclude file path	In this area, create a list of files to be skipped during coverage analysis. – To add an item to the list, click the Add button + and choose the required file in the dialog that opens . – To delete a file from the list so coverage is measured for it, select the file and click the Remove button –.

Toolbar

ItemShortcutDescription		
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

ItemDescription	
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

ItemKeyboardDescription	shortcut	
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.

	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Karma

In this dialog box, create configurations for running and debugging JavaScript unit tests using the Karma test runner.

On this page:

- Getting access to the Run/Debug Configuration: Karma dialog
- Karma-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Karma dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Using the Node Package Manager, install the Karma test runner as described in Preparing to Use Karma Test Runner .
4. Make sure the Karma plugin is activated. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Karma-specific configuration settings

Item	Description
Name	In this text box, specify the name of the run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Node.js interpreter	
Karma Node.js package	In this field, specify the Karma installation home `/npm/node_modules/karma` . If you installed Karma regularly through the Node Package Manager, PhpStorm detects the Karma installation home itself. Alternatively, type the path to executable file manually, or click the Browse button ... and select the location in the dialog box, that opens.
Configuration file	In this field, specify the location of the Karma configuration file . Normally, the file has the extensions `.conf.js`

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: − Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition.

For more information, see Configuring Third-Party Tools and External Tools .

- Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
  This option is available only if you have already at least one run/debug configuration in the current project.
- Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
- Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
  Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
  Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Meteor

Use this dialog box to create configurations for running and debugging Meteor applications.

The dialog box is available when the Meteor plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

On this page:

- Configuration tab
- Browser / Live Edit tab
- Toolbar
- Common options

Configuration tab

Item	Description
Name	In this text box, specify the name of the current run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Meteor Executable	In this field, specify the location of the Meteor executable file (see Installing Meteor).
Program Arguments	In this field, specify the command line additional parameters to be passed to the executable file on start up, if applicable. These can be, for example, `--dev` , `--test` , or `--prod` to indicate the environment in which the application is running (development, test, or production environments) so different resources are loaded on start up.
Working Directory	In this field, specify the folder under which the application files to run are stored. This folder must contain a `.meteor` folder in the root to be treated a Meteor project. By default, the field shows the path to the PhpStorm project root. Technically, several Meteor projects that implement different applications can be combined within one single PhpStorm project. To run and debug these applications independently, create a separate run configuration for each of them with the relevant working directory. To avoid port conflicts, these run configurations should use different ports. In the Program Arguments field, specify a separate port for each run configuration in the format `--port=<port_number>` .
Environment Variables	In this field, specify the environment variables for the Meteor executable file, if applicable.

Browser / Live Edit tab

In this tab, configure the behaviour of the browser and enable debugging the client-side code of the application. This functionality is provided through a `JavaScript Debug` run configuration, so technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated Meteor run configuration.

Item	Description
Open Browser	In the text box in this area, specify the URL address to open the application at. If you select the After Launch check box, the browser will open this page automatically after the application starts. Alternatively you can view the same result by opening the page with this URL address in the browser of your choice manually.
After Launch	Select this check box to have a browser started automatically after a debugging session is launched. Specify the browser to use in the drop-down list next to the check box. – To use the system default browser, choose Default. – To use a custom browser, choose it from the list. Note that Live Edit is fully supported only in Chrome. – To configure browsers, click the Browse button ... and adjust the settings in the Web Browsers dialog box that opens. For more information, see Configuring Browsers .
With JavaScript Debugger	Select this check box to enable debugging the client-side code in the selected browser.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
–	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` .

This check box is not available when editing the run/debug configuration defaults.

Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details. – Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Mocha

In this dialog box, create configurations for running and debugging JavaScript unit tests using the Mocha test framework.

On this page:

- Getting access to the Run/Debug Configuration: Mocha dialog
- Mocha-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Mocha dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Using the Node Package Manager, install the Mocha test framework as described in Preparing to Use Mocha Test Framework .

Mocha-specific configuration settings

Item	Description
Node.js interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
Node options	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. See Node Parameters for details.
Working directory	In this field, specify the working directory of the application. All references in test scripts will be resolved relative to this folder, unless such references use full paths. By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
Environment variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button –. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.
Mocha package	In this field, specify the Mocha installation home `/npm/node_modules/mocha` . If you installed Mocha regularly through the Node Package Manager, PhpStorm detects the Mocha installation home itself. Alternatively, type the path to executable file manually, or click the Browse button ... and select the location in the dialog box, that opens.
User interface	From this drop-down list, choose the interface according to which the tests in the test folder are written. PhpStorm will recognize only tests that comply with the chosen interface. If during test execution PhpStorm runs against a test of another interface, the test session will be canceled with an error. This means that all the tests in the specified test folder must be written according to the same interface and this interface must be chosen from the drop-down list.
Tests	In this area, specify the tests to be executed. The available options are: – All in directory: choose this option to run all the tests from files stored in a folder. In the Test directory field, specify the folder with the tests. To have PhpStorm look for tests in the subfolders under the specified directory, select the Include subdirectories check box. – File pattern: choose this option to have PhpStorm look for tests in all the files with the names that match a certain mask and specify the mask in the Test file pattern field. – Test file: choose this option to run only the tests from one file and specify the path to this file in the Test file field. – Suite: choose this option to run individual suites from a test file. In the Test file field, specify the file where the required suites are defined. Click the Suite name field and configure a list of suites to run. To add a suite to the list, click + and type the name of the required suite. To remove a suite, select it in the list and click – – Test: choose this option to run individual tests from a test file. In the Test file field, specify the file where the required tests are defined. Click the Test name field and configure a list of tests to run. To add a test to the list, click + and type the name of the required test. To remove a test, select it in the list and click –

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.

Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

−	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Node JS

In this dialog box, create configurations for running and debugging of NodeJS applications locally. "Locally" in the current context means that PhpStorm itself starts the NodeJS runtime environment installed on your computer, whereupon initiates a running or debugging session.

On this page:

- Getting access to the Run/Debug Configuration: NodeJS dialog
- Configuration tab
- Browser / Live Edit tab
- V8 Profiling tab
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: NodeJS dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .

Configuration tab

Item	Description
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
Node Parameters	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. The acceptable options are:
	`--debug-brk==<port for connect to debugger remotely>` – With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start. – With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later. — Specify this option to enable remote debugging of the application without re-starting the NodeJS server.
	`-v, --version` — Print the current version of NodeJS.
	`-e, --eval script` — Evaluate script.
	`--v8-options` — Print v8 command line options.
	`--vars` — Print various compiled-in variables.
	`--max-stack-size=val` — Set max v8 stack size (bytes).
	`--require coffee-script/register` — Specify this parameter to have CoffeeScript files transpiled into JavaScript on the fly during run. This mode requires that the `register.js` file, which is a part of the `coffee-script` package, should be located inside the project. Therefore you need to install the `coffee-script` package on the Node.js page locally, as described in Installing and Removing External Software Using Node Package Manager .
Working Directory	In this field, specify the working directory of the application. All references in the starting NodeJS application file, for example, imports, will be resolved relative to this folder, unless such references use full paths. By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
JavaScript File	In this field, specify the full path to the file to start running or debugging the application from. If you are going to debug CoffeeScript, specify the path to the generated JavaScript file with source maps. The file can be generated externally or through transpilation using file watchers. For more details, see Transpiling CoffeeScript to JavaScript .
Application Parameters	In this text box, type the NodeJS-specific arguments to be passed to the application start file through the process.argv array.
Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.
Path Mappings	

Browser / Live Edit tab

In this tab, configure the behaviour of the browser and enable debugging the client-side code of the application. This functionality is provided through a `JavaScript Debug` run configuration, so technically, PhpStorm creates separate run configurations for the server-side and the client-side code, but you specify all your settings in one dedicated NodeJS run configuration.

However, activating Live Edit in this tab lets you invoke Live Edit without creating a separate configuration, during a debugging session launched through the Node.js configuration.

For more information, see Live Editing of HTML, CSS, and JavaScript .

Item	Description
Open Browser	In the text box in this area, specify the URL address to open the application at. If you select the After Launch check box, the browser will open this page automatically after the application starts. Alternatively you can view the same result by opening the page with this URL address in the browser of your choice manually.
After launch	Select this check box to have a browser started automatically after a debugging session is launched. Specify the browser to use in the drop-down list next to the check box. – To use the system default browser, choose Default. – To use a custom browser, choose it from the list. Note that Live Edit is fully supported only in Chrome. – To configure browsers, click the Browse button ... and adjust the settings in the Web Browsers dialog box that opens. For more information, see Configuring Browsers .
With JavaScript debugger	Select this check box to enable debugging the client-side code in the selected browser.

V8 Profiling tab

Item	Description
Record CPU profiling info	Select this check box to start logging the CPU profiling data when the application is launched. The controls in the area below become enabled. Specify the following: – Log folder: in this field, specify the folder to store recorded logs in. Profiling data are stored in V8 log files `isolate-<session number>` . – One log file for isolates: – Tick package: in this field, specify the tick package to use. Choose the relevant package from the Tick package drop-down list or click the ... button next to it and choose the package in the dialog box that opens. – Gnuplot package: in this field, specify the location of the Gnuplot executable file to explore a timeline view that shows where V8 is spending time.
Allow taking heap snapshots	Select this check box to... The controls in the area below become enabled. Specify the following: – V8 profiler package: in this field, specify the v8-profiler package to use. Choose the relevant package from the v8-profiler package drop-down list or click the ... button next to it and choose the package in the dialog box that opens. – Communication port: in this field, specify the port through which PhpStorm communicates with the profiler, namely, sends a command to take a snapshot when you click the Take Heap Snapshot button on the toolbar of the Run tool window.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
–	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	`Alt+Insert`	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project.

- Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
- Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Node.js Interpreters Dialog

File | Settings | Languages and Frameworks | Node.js and NPM for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Node.js and NPM for OS X

The dialog box opens when you click the Browse button next to the Node Interpreter drop-down list in the Run/Debug Configuration: Node JS dialog or on the Node.js and NPM page.

Use this dialog box to configure Node.js installations as local and remote interpreters.

The term local Node.js interpreter denotes a Node.js installation on your computer. The term remote Node.js interpreter denotes a Node.js installation on a remote host or in a virtual environment set up in a Vagrant instance. See Configuring Node.js Interpreters for details.

When the dialog box opens from the Node.js and NPM page, you can only configure local interpreters installed on your computer. When the dialog box is accessed from the Run/Debug Configuration: Node JS dialog, both local and remote interpreters can be configured.

ItemTooltipDescription		
Node.js Interpreters		The list shows all the configured Node.js interpreters, both local and remote ones. For local interpreters, PhpStorm also shows the path to the Node.js executable file and to the associated npm executable file, see Installing and Removing External Software Using Node Package Manager .
+	Add	Click this button to add a new Node.js interpreter to the list. From the drop-down menu, choose All Local or Add Remote. Note that the Add Remote item is available only you opened the dialog box from the Run/Debug Configuration: Node JS dialog. Depending on your choice, either select the relevant local Node.js installation or configure a remote interpreter in the Configure Node.js Remote Interpreter Dialog that opens.
−	Delete	Click this button to remove the selected interpreter from the list.
	Edit	Click this button to create a new interpreter with the settings copied from the selected one.

Configure Node.js Remote Interpreter Dialog

The dialog box is available only when the Node.js Remote Interpreter plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

The dialog box opens when you click the Add toolbar button + in the Node.js Interpreters Dialog and choose Remote... from the drop-down menu. This menu item is available only when you open the Node.js Interpreters Dialog from the Run/Debug Configuration: Node JS .

Use this dialog box to configure access to Node.js installations on remote hosts or in development environments set up in Vagrant instances.

Item	Description

SSH Credentials	Choose this option to configure access to a Node.js interpreter on a remote host through SSH credentials. In the fields of the dialog box, specify the following:
	– Host: in this field, type the name of the host on which the interpreter is installed.
	– Port: in this field, type the port which the SSH server on the remote host listens to. The default port number is 22.
	– User name: in the field, type the user name under which you are registered on the SSH server.
	– Auth type: from this drop-down list, choose the authentication method.
	  – To access the host through a password, choose Password from the Auth type drop-down list and type the password.
	  – To access the host through a pair of SSH keys, choose Key pair, then specify the path to the file where your private key is stored or the passphrase if you have configured it during the generation of the key pair.
	
	To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm first attempts to retrieve path mappings itself by processing all the available application-level configurations. If PhpStorm finds the configurations with the same host as the one specified above, in the Host field, the mappings from these configurations are merged automatically. If no configurations with this host are found, PhpStorm displays an error message informing you that path mappings are note configured.
	
	To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the server access configuration in question, switch to the Mappings tab, and map local folders to folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.

Vagrant	This option is available only when the Vagrant repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Choose this option to configure access to a Node.js interpreter installed in a Vagrant instance using your Vagrant credentials. Technically, it is the folder where the VagrantFile configuration file for the desired environment is located. Based on this setting, PhpStorm detects the Vagrant host and shows it as a link in the Vagrant Host URL read-only field. To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm evaluates path mappings from the `VagrantFile` configuration file.
Deployment Configuration	This option is available only when the Remote Hosts Access plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins . Choose this option to configure access to a Node.js interpreter on a remote host using a server access configuration. This option is available only if you have at least one server access configuration of the type SFTP, see Creating a Remote Server Configuration . From the Deployment Configuration drop-down list, choose the server access configuration of the type SFTP according to which you want PhpStorm to connect to the target host. If the settings specified in the chosen configuration ensure successful connection, PhpStorm displays the URL address of the target host as a link in the Deployment Host URL field. To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . By default, PhpStorm retrieves path mappings from the chosen server access (deployment) configuration. If the configuration does not contain path mappings, PhpStorm displays the corresponding error message. To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the relevant server access configuration, switch to the Mappings tab, and map the local folders to the folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.
Node.js Interpreter Path	In this field, specify the location of the Node.js executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/node` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the Node.js home directory must be open for edit. When you click OK, PhpStorm checks whether the Node.js executable is actually stored in the specified folder. – If no Node.js executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway. – If the Node.js executable is found, you return to the Node.js Interpreters where the installation folder and the detected version of the Node.js interpreter are displayed.

Run/Debug Configuration: Node JS Remote Debug

In this dialog box, create configurations for debugging already running Node.js applications. This approach gives you the possibility to re-start a debugging session without re-starting the NodeJS server.

On this page:

- Getting access to the Run/Debug Configuration: NodeJS Remote Debug dialog
- NodeJS Remote Debug-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: NodeJS Remote Debug dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .

NodeJS Remote Debug-specific configuration settings

Item	Description
Host	In this text box, specify the host where the application is running.
Debug port	In this text box, specify the port to connect to. Do one of the following: – Copy the port number from the information message in the Run tool window that controls the running application. – Copy the port number from the Node parameter text box in the Run/Debug Configuration:NodeJS dialog box.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.

- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: NodeUnit

In this dialog box, create configurations to run unit tests for NodeJS applications.

On this page:

- Getting access to the Run/Debug Configuration: NodeUnit dialog
- NodeUnit-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: NodeUnit dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .
3. Install the Nodeunit testing framework in one of the following ways:
   - Download the framework at https://github.com/caolan/nodeunit and install it according to the official instructions.
   - Install the `nodeunit` package using the Node Package Manager (NPM) as described in Installing and Removing External Software Using Node Package Manager .

NodeUnit-specific configuration settings

Item	Description
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the Browse button ... and select the location in the dialog box, that opens.
Working Directory	In this text box, specify the folder to find tests under. This can be the project root folder or the parent directory for the `test` folder. Type the path manually or click the Browse button ... and select the location in the dialog box, that opens.
Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.
Nodeunit Module	In this text box, specify the installation folder of the Nodeunit framework. Type the path manually or click the ... button and choose the folder in the dialog box that opens.
Run	From this drop-down list, choose the scope of tests to execute. The available options are: – All JavaScript test files in the directory: choose this option to to have PhpStorm run all the test files in a folder. In the Directory text box below, specify the path to the test folder relative to the working directory . – JavaScript test file: choose this option to have a specific test executed. In the JavaScript test file text box, type the path to the file relative to the working directory .

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources.

If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.

Before launch

Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Node-webkit

In this dialog box, create configurations for running and debugging JNode-webkit applications.

On this page:

– Node-Webkit-specific configuration settings

– Toolbar

– Common options

Node-Webkit-specific configuration settings

Item	Description
Node-webkit app	Specify the application to be launched. You can either select the entire directory that contains `package.json` file, or the same directory packed into an archive with the `.nw` extension. Use the following buttons: Shift+Enter — Click this button to find the desired application in the file system. Click this button to add macros, specified in the Path Variables page of the IDE settings.
Node-webkit arguments	Specify here the arguments to be passed to the application being launched. Use the following button: Shift+Enter — Click this button to open the textual editor.
Working directory	In this field, specify the working directory of the application. All references in test scripts will be resolved relative to this folder, unless such references use full paths. By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
Environment variables	In this field, specify the environment variables for the executable file, if applicable.
Node-webkit interpreter	Specify executable for the Node-webkit. Use the following buttons: Shift+Enter — Click this button to find the desired application in the file system. Click this button to add macros, specified in the Path Variables page of the IDE settings.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the

application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .

- Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
  This option is available only if you have already at least one run/debug configuration in the current project.
- Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
- Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
  Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
  Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: NPM

In this dialog box, create configurations for running npm scripts locally. "Locally" in the current context means that PhpStorm itself starts the NodeJS runtime environment installed on your computer, whereupon initiates script execution.

On this page:

- Getting access to the Run/Debug Configuration: NPM dialog
- Configuration tab
  - Toolbar
  - Common options

Getting access to the Run/Debug Configuration: NPM dialog

1. Install and enable the Node.js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. Download and install the NodeJS runtime environment that contains the Node Package Manager(npm) .

Configuration tab

Item	Description
Name	In this text box, specify the name of the run/debug configuration.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
package.json	In this field, specify the `package.json` file to run the scripts from. Choose the file from the drop-down list which shows all the `package.json` files detected in the current project or click ... and choose the required `package.json` in the dialog box that opens.
Command	From this drop-down list, choose the npm CLI command to execute, by default `run-script` is selected. Learn more at npm documentation , under the section CLI Commands.
Scripts	From this drop-down list, choose the script to which the chosen command will be applied. The list contains all the scripts defined within the `scripts` property in the `Package.json` file.
Arguments	In this field, specify the command line arguments to execute a script with. Learn more at Arguments .
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
Node Options	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. The acceptable options are:

`--debug-brk==<port for connect to debugger remotely>` – With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start. – With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later.	Specify this option to enable remote debugging of the application without re-starting the NodeJS server.
`-v, --version`	Print the current version of NodeJS.
`-e, --eval script`	Evaluate script.
`--v8-options`	Print v8 command line options.
`--vars`	Print various compiled-in variables.
`--max-stack-size=val`	Set max v8 stack size (bytes).
`--require coffee-script/register`	Specify this parameter to have CoffeeScript files transpiled into JavaScript on the fly during run. This mode requires that the `register.js` file, which is a part of the `coffee-script` package, should be located inside the project. Therefore you need to install the `coffee-script` package on the Node.js page locally, as described in Installing and Removing External Software Using Node Package Manager .

Environment Variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button –. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–		Click this button to remove the selected configuration from the list.

	Alt+Delete	
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PhoneGap/Cordova

Use this dialog box to create configurations for running and debugging applications developed through integration with the help of PhoneGap , Apache Cordova , and Ionic frameworks and intended for running on various mobile platforms, including Android .

On this page:

- Getting access to the Run/Debug Configuration: PhoneGap/Cordova dialog
- PhoneGap/Cordova/Ionic-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: PhoneGap/Cordova dialog

Make sure the PhoneGap/Cordova plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

PhoneGap/Cordova/Ionic-specific configuration settings

Item	Description
PhoneGap/Cordova Executable Path	In this field, specify the location of the executable file `phonegap.cmd` , `cordova.cmd` , or `ionic.cml` (see Installing PhoneGap/Cordova/Ionic).
PhoneGap/Cordova Working Directory	In this field, specify the folder under which the PhoneGap/Cordova/Ionic application files to run are stored.

Command	From this drop-down list, choose the command to run. The contents of the drop-down list, depend on the actually used framework, namely, on the executable file specified in the PhoneGap/Cordova Executable Path field. The available options are:
	– For PhoneGap:
	  – emulate
	  – run
	  – prepare
	  – serve
	  – remote build
	  – remote run
	
	  See https://www.npmjs.org/package/phonegap for a list of PhoneGap-specific commands with descriptions.
	– For Cordova:
	  – emulate
	  – run
	  – prepare
	  – serve
	
	  See https://www.npmjs.org/package/cordova for a list of Cordova-specific commands with descriptions.
	– For Ionic:
	  – emulate
	  – run
	  – prepare
	  – serve
	
	  See https://www.npmjs.org/package/ionic for a list of Ionic-specific commands with descriptions.

Platform	From this drop-down list, choose the platform for running on which the application is intended. The available options are: – Android – ios To emulate this platform, you need to install the ios-sim command line tool globally. You can do it through the Node Package Manager (npm), see Installing and Removing External Software Using Node Package Manager or by running the `sudo npm install ios-sim -g` command, depending on your oeprating system. – amazon-fireos – firefoxos – blackberry10 – ubuntu – wp8 – windows8 – browser Learn more about targeted platforms at http://docs.phonegap.com/en/edge/guide_platforms_index.md.html#Platform%20Guides and http://cordova.apache.org/docs/en/4.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface .
Specify Target	Select this check box to appoint an Android physical or virtual device to run the application on and select the required device from the drop-down list. The list shows all the virtual and physical devices that are currently configured on our machine. See http://docs.phonegap.com/en/edge/guide_platforms_android_index.md.html#Android%20Platform%20Guide for details. If PhpStorm displays the following error message: Cannot detect ios-sim in path, make sure you have installed the `ios-sim` , see Before you start.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.

	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	`Alt+Insert`	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

−	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHP Built-in Web Server

Use this dialog box to configure running PHP applications on a local built-in Web server.

Click here for the description of the options that are common for all run/debug configurations.

The dialog box consists of the following areas:

- Server Configuration area
- Command Line area
- Toolbar
- Common options

Server Configuration area

In this area, configure the access to the built-in Web server.

Item	Description
Host	In this text box, type `localhost`, because the built-in server is located on your machine.
Port	Use this spin box to specify the number of the port where the built-in web server runs. The port number should be set to 1024 or higher. By default this port is set to the default PhpStorm port through which PhpStorm accepts connections from services (`63342`).
Document root	In this text box, type the full path to the folder that will be considered server document root. This can be either the project root or any other folder under it. All the folders under the document root will be recursively submitted to the PHP interpreter. The server document root folder will be accessed through HTTP at the above specified host:port. URL addresses for other pages of your applications will be composed based on this mapping. Type the path manually or click the Browse button and choose the relevant folder in the dialog box, that opens.
Use router script	Select this check box to have a PHP router script executed at every HTTP request start-up. The script is run for each HTTP request. If this script returns `FALSE`, the requested resource is returned as-is. Otherwise the script output is returned to the browser. In the text box, specify the location of the script to run. Type the path manually or click the Browse button and choose the file in the dialog box, that opens.

Command Line area

In this area, customize the behavior of the current PHP interpreter by specifying the options and arguments to be passed to the PHP executable file.

Item	Description
Interpreter options	In this text box, specify the options to be passed to the PHP executable file. They override the default behavior of the PHP interpreter and/or ensure that additional activities are performed. If necessary, click and type the desired options in the Command Line Options dialog box. Type each option on a new line. When you close the dialog box, they are all displayed in the Command line options text box with spaces as separators.
Custom working directory	In this text box, specify the location of the files that are outside the folder with your sources and are referenced through relative paths. Type the path manually or click the Browse button and select the desired folder in the dialog that opens.
Environment variables	In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
−	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place.

This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources.

If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.

Before launch

Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details. – Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHP HTTP Request

Use this dialog box to configure running and debugging of separate HTTP Requests . This is helpful when you are actually interested in a specific page that is accessed in a number of steps, but for this or that reason you cannot specify this page as the start page for debugging, for example, because you need to "come" to this page with certain data. For details, see Debugging a PHP HTTP Request .

The dialog box consists of the following areas:

- Configuration
- Toolbar
- Common options

Click here for the description of the options that are common for all run/debug configurations.

Configuration

Item	Description
Server	From this drop-down list, select the server access configuration to interact with the Web server where the application is executed.
URL	In this text box, specify the `host` element of the request in question. Type the element relative to the host specified in the debug server configuration, see Creating a PHP Debug Server Configuration . As you type, PhpStorm composes the URL address on-the-fly and displays it below the text box.
Request method	From this drop-down list, choose the relevant request type. The available options are: – POST – GET
Query String	In this text box, type the query string of the request, this string will be appended to the request after the `?` symbol.
Request Body	In this text box, type the data to be sent to the server through the POST request. The text box is available only for request methods of the type POST. By default, the Project Encoding is used in requests encoding if it is not specified explicitly, for example: `header('Content-type: text/html;charset=utf-8');` The Project Encoding is specified on the File Encodings page, under the Editor node of the Settings / Preferences Dialog .

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens,

select the configuration to run.
This option is available only if you have already at least one run/debug configuration in the current project.

- Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
- Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHP Remote Debug

Use this dialog box to configure debugging of PHP applications on a remote server.

The dialog box consists of the following areas:

- Configuration
- Toolbar
- Common options

Click here for the description of the options that are common for all run/debug configurations.

Configuration

Item	Description
Server	Use this drop-down list to specify the Web server configuration to use. The list shows all the configurations that are currently available in PhpStorm.
...	Click this button to open the Deployment page and view the details of the selected configuration there.
Ide key (session id)	In this text box, specify the key to identify the debugging session.
Debug	Use the controls in this area to configure behaviour of the debugging tool. – Break at the first line - select this check box to have the debugging tool stop at the first line of the source code.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`. This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you

made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:

- If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
- If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.

- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

—	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHP Script

Use this dialog box to configure running and debugging of single PHP files locally using a PHP console.

Click here for the description of the options that are common for all run/debug configurations.

The dialog box consists of the following areas:

- Configuration area
- Command Line area
- Debug area
- Toolbar
- Common options

Configuration area

In this area, specify the script to run or debug and the parameters to process it with, if applicable.

Item	Description
File	In this text box, specify the location of the file to run or debug. Type the path to the file manually or click the Browse button and select the desired location in the Choose PHP File dialog box that opens.
Arguments	In this text box, type the list of arguments to be passed to the PHP script, same way as if you were entering these parameters in the command line. If necessary, click and type the desired switches in the Arguments dialog box. Type each argument on a new line. As you type, they appear in the Arguments text box with spaces as separators.

Command Line area

In this area, customize the behavior of the current PHP interpreter by specifying the options and arguments to be passed to the PHP executable file.

Item	Description
Interpreter options	In this text box, specify the options to be passed to the PHP executable file. They override the default behavior of the PHP interpreter and/or ensure that additional activities are performed. If necessary, click and type the desired options in the Command Line Options dialog box. Type each option on a new line. When you close the dialog box, they are all displayed in the Command line options text box with spaces as separators.
Custom working directory	In this text box, specify the location of the files that are outside the folder with the script and are referenced in your script through relative paths. Type the path manually or click the Browse button and select the desired folder in the dialog that opens . This setting does not block the script execution because the script location is always specified through a full path.
Environment variables	In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Debug area

Use the controls in this area to configure behaviour of the debugging tool.

Item	Description
Break at the first line	Select this check box to have the debugging tool stop at the first line of the source code.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
−	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.

Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHPUnit

Use this dialog box to create a configuration to be used for running and debugging unit tests on PHP applications in the console using the PHPUnit framework. Click here for the description of the options that are common for all run/debug configurations.

On this page:

- Before you start
- Test Runner area
- Command Line area
- Toolbar
- Common options

Before you start

To run PHPUnit tests:

1. Install and configure the PHPUnit tool on your computer, see http://www.phpunit.de/manual/current/en/installation.html .
2. Make sure the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Test Runner area

In this area, specify the unit tests to launch and the command line switches to be passed to the test runner.

Item	Description
Test scope	In this area, specify the location of tests or the configuration file where they are listed. – Directory: select this option to have all the unit tests in a directory launched. In the Directory text box, specify the directory to search the unit test in. Type the path to the directory manually or click the Browse button and select the desired directory in the Choose Test Directory dialog box, that opens. – Class: select this option to have all the unit tests in a test class launched. 1. In the File text box, specify the file to search the class in. Type the path to the file manually or click the Browse button and select the desired directory in the Choose Test File dialog box, that opens. 2. In the Class text box, specify the desired class. Type the class name manually or click the Browse button and select the desired class in the tree view, that opens. – Method: select this option to have a specific test method launched. 1. In the File text box, specify the file to search for the test method in. Type the file name manually or click the Browse button and select the desired file in the tree view, that opens. 2. In the Method text box, specify the desired method. – Defined in the configuration file: select this option to have test runner execute the tests from a dedicated XML file. – To use the default configuration file specified on the PHPUnit page of the Settings dialog box, clear the Use alternative configuration file check box. If no default configuration file is appointed on the PHPUnit page, the run/debug configuration is invalid. – To run the tests from a custom configuration file, select the Use alternative configuration file check box and specify the location of the file to use in the text box.
Test runner options	In this text box, specify the test runner switches . If necessary, click and type the desired switches in the Command Line Options dialog box. Type each switch on a new line. When you close the dialog box, the specified switches are displayed in the Test runner options text box with spaces as separators.

Command Line area

In this area, customize the behavior of the current PHP interpreter by specifying the options and arguments to be passed to the PHP executable file.

Item	Description
Interpreter options	In this text box, specify the options to be passed to the PHP executable file. They override the default behavior of the PHP interpreter and/or ensure that additional activities are performed. If necessary, click and type the desired options in the Command Line Options dialog box. Type each option on a new line. When you close the dialog box, they are all displayed in the Command line options text box with spaces as separators.
Custom working directory	In this text box, specify the location of the files that are outside the folder with tests and are referenced in your tests through relative paths. Type the path manually or click the Browse button and select the desired folder in the dialog that opens . This settings does not block the test execution because the location of tests is always specified through a full path to the corresponding files and/or directories.
Environment variables	In this field, specify the environment variables be passed to the built-in server. See Environment Variables in Apache for details.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHPUnit by HTTP

Use this dialog box to configure running unit tests of PHP applications on a remote server./p> Click here for the description of the options that are common for all run/debug configurations.

On this page:

- Before you start
- Configuration tab
- Test Groups tab
- Remote tab
- Toolbar
- Common options

Before you start

To run PHPUnit tests:

1. Install and configure the PHPUnit tool on your computer, see http://www.phpunit.de/manual/current/en/installation.html .
2. Make sure the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Configuration tab

In this tab, specify the unit tests to launch and the command line switches to be passed to the test runner.

Item	Description
Test	In this area, specify the location of tests or the configuration file where they are listed. – All in Directory: select this option to have all the unit tests in a directory launched. In the Directory text box, specify the directory to search the unit test in. Type the path to the directory manually or click the Browse button ... and select the desired directory in the Choose Test Directory dialog box, that opens. – All in File: select this option to have all the unit tests in a test class or a file launched. In the File text box, specify the file to search the class or suite in. Type the path to the file manually or click the Browse button ... and select the desired directory in the Choose Test File dialog box, that opens. – Class or Suite: select this option to have all the unit tests in a test class or a test suite launched. 1. In the File text box, specify the file to search the class or suite in. Type the path to the file manually or click the Browse button ... and select the desired directory in the Choose Test File dialog box, that opens. 2. In the Class text box, specify the desired class. Type the class name manually or click the Browse button ... and select the desired class in the tree view, that opens. – Method: select this option to have a specific test method launched. 1. In the File text box, specify the file to search for the test method in. Type the file name manually or click the Browse button ... and select the desired file in the tree view, that opens. 2. In the Method text box, specify the desired method. – XML file: select this option to have the test runner execute the tests from an XML configuration file. – To use the default configuration file specified on the PHPUnit page of the Settings dialog box, clear the Use alternative configuration file check box. If no default configuration file is appointed on the PHPUnit page, the run/debug configuration is invalid. – To run the tests from a custom configuration file, select the Use alternative configuration file check box and specify the location of the file to use in the text box.
Test Runner options	
Custom working directory	In this text box, specify the location of the files that are outside the folder with tests and are referenced in your tests through relative paths. Type the path manually or click the Browse button ... and select the desired folder in the dialog that opens . This setting does not block the test execution because the location of tests is always specified through a full path to the corresponding files and/or directories.

Test Groups tab

In this tab, appoint the groups of tests to execute when the tests are run according to the current run configuration.

Grouping tests is helpful, for example, to distinguish between tests to run in a production environment from those to run in your development environment. You just need to create two groups and then include or exclude them depending on the current environment.

To attach a test to a group, tag it with the @group annotation in the format:

```
/**
@group <group specification>
*/
```

To enable filtering tests based on their authors, tag the tests with the @author annotation.

Item	Description
Include/Exclude groups	Select this check box to enable configuring execution of test groups .
Group Name	In this read-only list, select the test group to be involved or skipped in testing.
Include	Select this check box to have the tests from the selected test group executed.
Exclude	Select this check box to have the tests from the selected test group skipped.

Remote tab

In this tab, configure deployment of tests to a remote server.

Item	Description
Debug server	In this field, specify the configuration through which the debugging engine will interact with the server through HTTP. Actually, a configuration consist of a URL address of the target environment, the debugger type, and possibly mappings between folders in your project and their correspondences on the server. Choose a configuration from the drop-down list or click the Browse button ... and define a new configuration in the Servers dialog box that opens.
Upload test directory before run	– When this check box is selected, PhpStorm automatically uploads tests to the default server according to the deployment options. – When the check box is cleared, you have to upload tests manually, see Uploading and Downloading Files . The check box is by default selected.
Remove test files after run	– When this check box is selected, PhpStorm automatically removes the executed tests from the server upon execution. – When the check box is cleared, tests remain on the server after execution and you will have to download them manually, see Uploading and Downloading Files .
Show transfer logs	Select this check box to have test deployment logged.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
−	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	`Alt+Insert`	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated.

In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .

- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: PHP Web Application

Use this dialog box to configure running and debugging of PHP applications on a remote server. According to the settings of this configuration, PhpStorm fully controls the debugging process: it launches the application, opens the browser, and activates the debugging engine. A PHP Web Application debug configuration tells PhpStorm the URL address to access the starting page of the application, the browser to open the starting page in, and the debug server configuration to use. For details, see Debugging with a PHP Web Application Debug Configuration .

Click here for the description of the options that are common for all run/debug configurations.

The dialog box consists of the following areas:

- Configuration area
- Toolbar
- Common options

Configuration area

Item	Description
Server	Use this drop-down list to specify the Web server configuration to use. The list shows all the configurations that are currently available in PhpStorm.
...	Click this button to open the Servers page and view the details of the selected configuration there.
Start URL	In this area, compose the URL address to access the application through. In the Start URL text box, specify the local file that implements the starting page of the application. Type the path to the desired file relative to the folder that is mapped to the root folder of the target host. PhpStorm concatenates the host root URL with the specified relative path and shows the URL address of the application starting page in the read-only field below.
Browser	From this drop-down list, select the Web browser to open the application in.
Debug	Use the controls in this area to configure behaviour of the debugging tool. – Break at the first line - select this check box to have the debugging tool stop at the first line of the source code.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
or	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.

- Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
- Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
  Specify the location of the Node.js interpreter and the parameters to pass to it.
- Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
  - If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
  - If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
- Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
- Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
- Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
- Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

	`Alt+Delete`	Click this icon to remove the selected task from the list.
	`Enter`	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	`Alt+Up`	Click this icon to move the selected task one line up in the list.
	`Alt+Down`	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Spy-js

In this dialog box, create configurations for tracing Web applications using the Spy-js tool.

On this page:

- Getting access to the Run/Debug Configuration: Spy-js dialog
- Spy-js-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Spy-js dialog

1. Download and install Node.js because it is used by the Spy-js trace server.
2. Install and enable the Spy-js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Spy-js-specific configuration settings

Item	Description
Node interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file. With Spy-js, it is recommended to use Node.js version 0.10.24 or higher.
Trace server port	In this spin box, specify the port on which Spy-js will act as a proxy server. This port number must be the same as your system proxy port. If the Automatically configure system proxy check box is selected, the specified port number is automatically set for the system proxy server. Otherwise you will have to specify the value of the field in the system proxy settings manually. The trace server port is filled in automatically. To avoid port conflicts, it is recommended that you accept the suggested value and keep the Automatically configure system proxy check box selected.
Use	In this drop-down list, specify the way to configure a tracing session. – To have Spy-js apply its internal predefined configuration, choose Default configuration. – To have your custom manually created configuration applied, choose the Configuration file option and then specify the location of your custom configuration file in the Configuration field below. A configuration file is a JavaScript file with the extension `.js` or `.conf.js` that contains valid JavaScript code that meets the Spy-js configuration requirements . If PhpStorm detects files with the extension `.conf.js` in the project, these files are displayed in the drop-down list. Type the path to the configuration file manually or click the Browse button ... and choose the location in the dialog box that opens. Once specified, a configuration file is added to the drop-down list so you can get if next time from the list instead of specifying the path.
URL to trace	In this field, specify the URL address of the Web page to capture events on. By default, the field is empty. This means that Spy-js captures events on all the currently opened Web pages. If you want to restrict the tracing to a certain page, specify its URL address. Type the address manually or choose it from the drop-down list, if it has been once specified in the configuration.
Automatically configure system proxy	– When this check box is selected, the system proxy server is activated automatically with the port specified in the Trace server port field. – Clear this check box to specify proxy settings manually. See how to configure proxy settings manually on Windows , Mac , Ubuntu , iOS , Android , Windows Phone . Please note that some desktop browsers have their own screens for proxy settings configuration. The check box is selected by default, and it is strongly recommended that you accept this setting and have the proxy configured automatically.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
–	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place.

This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources.

If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.

Before launch

Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description
+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added: – Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools . – Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run. This option is available only if you have already at least one run/debug configuration in the current project. – Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details. – Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package. – Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool. Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package. – Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with. Specify the location of the Node.js interpreter and the parameters to pass to it. – Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected: – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start. – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched. – Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support . – Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens. – Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details. – Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .
–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
↑	Alt+Up	Click this icon to move the selected task one line up in the list.
↓	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: Spy-js for Node.js

In this dialog box, create configurations for tracing Node.js applications using the Spy-js tool.

On this page:

- Getting access to the Run/Debug Configuration: Spy-js for Node.js dialog
- Spy-js for Node.js-specific configuration settings
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: Spy-js for Node.js dialog

1. Download and install Node.js because it is used by the Spy-js trace server.
2. Install and enable the Spy-js plugin. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Spy-js for Node.js-specific configuration settings

Item	Description
Node interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file. With Spy-js, it is recommended to use Node.js version 0.10.24 or higher. To enable tracing ECMAScript 6 scripts, use Node.js version 0.11.13 or higher and specify `--harmony` as a Node parameter.
Node parameters	In this text box, type the NodeJS-specific command line options to be passed to the NodeJS executable file. For example, to enable tracing ECMAScript 6 scripts, specify `--harmony` as a Node parameter. Note that Node.js must be version 0.11.13 or higher.
Working directory	In this field, specify the working directory of the application. All references in the starting NodeJS application file, for example, imports, will be resolved relative to this folder, unless such references use full paths. By default, the field shows the project root folder. To change this predefined setting, choose the desired folder from the drop-down list, or type the path manually, or click the Browse button ... and select the location in the dialog box, that opens.
JavaScript file	In this field, specify the full path to the file to start running the application from. If you are going to trace CoffeeScript, specify the path to the generated JavaScript file. The file can be generated externally or through transpilation using file watchers. For more details, see Transpiling CoffeeScript to JavaScript .
Application parameters	In this text box, type the NodeJS-specific arguments to be passed to the application start file through the process.argv array.
Environment variables	In this field, specify the environment variables for the NodeJS executable file, if applicable. Click the Browse button ... to the right of the field and configure a list of variables in the Environment Variables dialog box, that opens: – To define a new variable, click the Add toolbar button + and specify the variable name and value. – To discard a variable definition, select it in the list and click the Delete toolbar button —. – Click OK, when ready The definitions of variables are displayed in the Environment variables read-only field with semicolons as separators. The acceptable variables are: – `NODE_PATH` : A `:` -separated list of directories prefixed to the module search path. – `NODE_MODULE_CONTEXTS` : Set to 1 to load modules in their own global contexts. – `NODE_DISABLE_COLORS` : Set to 1 to disable colors in the REPL.
Configuration file	From this drop-down list, choose the file with the configuration settings to apply to the tracing session. A configuration file is a JavaScript file with the extension `.js` or `.conf.js` that contains valid JavaScript code that meets the Spy-js configuration requirements . If PhpStorm detects files with the extension `.conf.js` in the project, these files are displayed in the drop-down list. Type the path to the configuration file manually or click the Browse button ... and choose the location in the dialog box that opens. Once specified, a configuration file is added to the drop-down list so you can get if next time from the list instead of specifying the path.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to add a new configuration to the list.
—	Alt+Delete	Click this button to remove the selected configuration from the list.
	Ctrl+D	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	Alt+Up or Alt+Down	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.

Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	**Keyboard shortcut**	**Description**

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

–	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Run/Debug Configuration: XSLT

This run/debug configuration becomes available, when XPath View+XSLT Support plugin is enabled.

On this page

- Getting access to the Run/Debug Configuration: XSLT dialog
- Settings tab
- Advanced tab
- Toolbar
- Common options

Getting access to the Run/Debug Configuration: XSLT dialog

Make sure the XPath View+XSLT Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Settings tab

Item	Description
Input	Use the controls in this area to specify the XML file to process and the script to be executed. – XSLT Script File - in this text box, specify the path to the XSLT stylesheet file. Type the path manually or click the Browse button ... and select the desired file in the Choose XSLT File dialog box, that opens. – Choose XML Input File - from this drop-down list, select the XML input file to be transformed. The list contains all the XML files that have been associated with the chosen stylesheet via the File Associations functionality. To specify a file, which is not on the list, click the Browse button ... and select the desired file in the Choose XML File dialog box, that opens.

Output	Use the controls in this area to configure handling of the script output.
	– Show in Default Console - select this option to have the output displayed in the normal run console, together with any warnings and error messages from the XSLT transformer, as well as messages generated by the script, e.g. by `xsl:message` .
	– Show in Extra Console Tab - select this option to have the produced output displayed in an extra, XSLT Output, tab.
	This option is selected by default.
	– Highlight Output As - from this drop-down list, select the file type to highlight the output as.
	– Save to File - select this option to have the output saved directly to a file. In the text box, specify the name of the target file. Type the path to the file manually or click the Browse button ... and select the desired file in the Choose Output File dialog box, that opens. If you type the name of a file that does not exist, PhpStorm will create a file and save the output to it.
	  – Open File in Editor After Execution - select this check box to have the file with the output opened in the editor after the script is executed successfully.
	  – Open File in Web Browser After Execution - select this check box to have the file with the output opened in the configured Web browser after the script is executed successfully.
	Warning! The specified file will be overwritten without requesting for confirmation.

Parameters	Use the controls in this area to create and manage a list of parameters to be passed to the script. – Add (+) - click this button to create a new entry. – Remove (—) - click this button to remove the selected entry from the list. – Name - in this text box, specify the name of the parameter. – Value - in this text box, specify the value of the parameter. Warning! The field is mandatory. Parameters without values are not passed to the script. Values are not assigned by default.

Advanced tab

In this tab, configure additional options that are not commonly required in run configurations.

Item	Description
Smart Error Handling	– Clear this check box to have the console display full error messages including their complete stack traces, when an error occurs during execution. – Select this check box to suppress showing staktraces and have the console display only the relevant information about errors.
VM Arguments	In this text box, specify optional VM arguments to be passed to the VM where the XSLT script is executed. These can be heap size, garbage collection options, file encoding, etc. If the line of VM arguments is too long, click the button and and type the text in the VM Arguments dialog box, that opens.
Working Directory	In this text box, specify the working directory to use. Type the path manually or click the Browse button ... and select the desired folder in the Working Directory dialog box, that opens. If no folder is specified in text box, the working directory will be the one where the XSLT script file is located.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new configuration to the list.
—	`Alt+Delete`	Click this button to remove the selected configuration from the list.
	`Ctrl+D`	Click this button to create a copy of the selected configuration.
	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for newly created configurations.
↑ or ↓	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or folder up and down in the list. The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar.
	Sort configurations	Click this button to sort configurations in alphabetical order.

Common options

Item	Description
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Temporary configurations limit	Specify here the maximum number of temporary configurations to be stored and shown in the Select Run/Debug Configuration drop-down list.
Share	Select this check box to make the run/debug configuration available to other team members. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` . This check box is not available when editing the run/debug configuration defaults.
Single instance only	If this check box is selected, this run/debug configuration cannot be launched more than once. Every time a new run/debug configuration is launched, PhpStorm checks the presence of the other instances of the same run/debug configuration, and displays a confirmation dialog box. If you click OK in the confirmation dialog box, the first instance of the runner will be stopped, and the next one will take its place. This makes sense when the usage of certain resources can cause conflicts, or when launching two run/debug configurations of the same type consumes too much of the CPU and memory resources. If this check box is not selected, it is possible to launch as many instances of the runner as required. So doing, each runner will start in its own tab of the Run tool window.
Before launch	Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.

Item	Keyboard shortcut	Description

+	Alt+Insert	Click this icon to add a task to the list. Select the task to be added:
		– Run External tool. Select this option to run an application which is external to PhpStorm. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in PhpStorm yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools .
		– Run Another Configuration. Select this option to have another run/debug configuration executed. In the dialog that opens, select the configuration to run.
		This option is available only if you have already at least one run/debug configuration in the current project.
		– Run File Watchers. Select this option to have PhpStorm apply all the currently active file watchers, see Using File Watchers for details.
		– Run Grunt task. Select this option to run a Grunt task. In the Grunt task dialog box that opens, specify the `Gruntfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Grunt tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `grunt-cli` package.
		– Run Gulp task. Select this option to run a Grunt task. In the Gulp task dialog box that opens, specify the `Gulpfile.js` where the required task is defined, select the task to execute, and specify the arguments to pass to the Gulp tool.
		Specify the location of the Node.js interpreter, the parameters to pass to it, and the path to the `gulp` package.
		– Run npm Script. Select this check box to execute an npm script. In the NPM Script dialog box that opens, specify the `package.json` file where the required script is defined, select the script to execute, choose the command to apply to it, and specify the arguments to execute the script with.
		Specify the location of the Node.js interpreter and the parameters to pass to it.
		– Compile TypeScript. Select this option to run the built-in TypeScript compiler and thus make sure that all the changes you made to your TypeScript code are reflected in the generated JavaScript files. In the TypeScript Compile Settings dialog that opens, select or clear the Check errors check box to configure the behaviour of the compiler in case any errors are detected:
		  – If the Check errors check box is selected, the compiler will show all the errors and the run configuration will not start.
		  – If the Check errors check box is cleared, the compiler will show all the detected errors but the run configuration still will be launched.
		– Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support .
		– Run Phing target: Add this task to execute the specified Phing target prior to running or debugging. To appoint a Phing target, click the Browse button ... and select the desired target in the Choose Phing Target to Execute dialog box, that opens.
		– Run Remote External tool: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
		– Upload files to Remote Host. Select this option to have the application files automatically uploaded to the server according to the default server access configuration. For more information, see Configuring Synchronization with a Web Server and Uploading and Downloading Files .

−	Alt+Delete	Click this icon to remove the selected task from the list.
	Enter	Click this icon to edit the selected task. Make the necessary changes in the dialog that opens.
	Alt+Up	Click this icon to move the selected task one line up in the list.
	Alt+Down	Click this icon to move the selected task one line down in the list.
Show this page		Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration.
Active tool window		Select this option if you want the Run / Debug tool windows to be activated automatically when you run/debug your application. This option is enabled by default.

Save Project as Template Dialog

Tools | Save Project As Template

Specify the settings for the template that you are creating.

Item	Description
Name	Specify the name of the template.
Description	Type the template description. You can enclose text fragments between `<b>` and `</b>`, and `<i>` and `</i>` to make the corresponding fragments bold or italicized. For example: `A <b>PHP</b> project with a <i>Drupal</i> module.`

Select Path Dialog

Use this dialog to select the necessary files or folders.

The dialog name and the available functions depend on the task you are performing at the moment (inappropriate functions are normally disabled). For example, depending on the situation, you may be able to select only one item, or a number of items. There may be cases when you can select a folder or folders but cannot select a file or files, etc.

– Main functions

– Path field

– Using drag-and-drop from a file browser

Main functions

Most of the functions available in this dialog are accessed by means of the toolbar icons (shown in the Icon column). Alternatively, you can use context menu commands (accessed by right-clicking items in the tree; listed in the Command column) or keyboard shortcuts (the Shortcut column).

Icon	Command	Shortcut	Description
	Home	Ctrl+1	Use this icon, command or shortcut to select your home directory. For example, on Windows, this may be `C:\Users\<your_name>` .
	Desktop	Ctrl+D	Use this icon, command or shortcut to select the Desktop directory
	Project	Ctrl+2	Use this icon, command or shortcut to select your project root directory.
	New Folder	Alt+Insert	Use this icon, command or shortcut to create a new folder in the selected folder.
	Delete	Delete	Use this icon, command or shortcut to delete the selected file or folder.
	Refresh	Ctrl+Alt+Y	Use this icon, command or shortcut to synchronize the tree with the current state of the file system. (Under certain circumstances, PhpStorm may not be aware of the changes made externally unless you use this command.)
	Show or Hide Hidden Files and Folders		Use this icon or command to turn showing hidden files and folders on or off.
	Hide or Show path	Ctrl+P	Use this command or shortcut to hide or show the path field . (The command is located on the toolbar in the right-hand part of the dialog and is shown as a hyperlink.)

Path field

The path field (if not hidden) is located underneath the toolbar. This field shows the path to the item selected in the tree.

By using the path auto-completion feature available in this field, you can quickly navigate through the file system to select the necessary file or folder.

To activate path auto-completion, place the cursor in the field and press Ctrl+Space . Start typing. A pop-up will appear showing the contents of the current directory. Select an item in the pop-up. Continue typing and selecting until the necessary item is selected.

Use the button to the right to show the history list of recent entries.

Using drag-and-drop from a file browser

You can quickly locate and select the necessary file or folder if you drag the corresponding item from your file browser (Explorer, Finder, etc.) into the area where the tree is shown.

Specify Code Cleanup Scope Dialog

Code | Code Cleanup

Specify the scope for code cleanup. (Code cleanup means finding potentially problematic code fragments and automatically fixing them right away.)

Item	Description
Whole project	Select this option if you want to perform code cleanup for the whole project.
Uncommitted files	This option is only available for projects under version control. Select this option if you only want to perform code cleanup for the files that have not yet been committed to a version control system, and choose a changelist from the drop-down list.
File <file path>	Select this option to perform code cleanup for the file that is open in the editor or selected in the Project tool window.
Directory <directory path>	Select this option to perform code cleanup for the directory currently selected in the Project tool window.
Selected files	Select this option to perform code cleanup for the files selected in the Project tool window.
Custom scope	Select this option to specify a custom scope. Select one of the predefined scopes from the drop-down list, or click ... and define the scope in the Scopes dialog that opens. For instructions on how to define a scope, refer to Scope Language Syntax Reference .
Inspection profile	Select the inspection profile to be used. Choose a pre-defined profile from the drop-down list, or click ... and configure a profile in the Inspections dialog that opens. You can open the Inspections dialog to check which fixes will be applied to the selected scope when you run code cleanup.

Specify Code Duplication Analysis Scope

Code | Locate Duplicates

Use this dialog box to launch the search for duplicated code fragments in the specified scope.

Item	Description
Whole project	Select this option to perform the analysis for the whole project.
Module <name>	Select this option to have PhpStorm analyze the module that is currently selected in the Project tool window.
File <name>	Select this option to have PhpStorm analyze the file that is currently selected in the Project tool window or opened in the editor.
Custom scope	Select this option to define a custom scope to apply analysis to. Choose a pre-defined scope from the drop-down list or click the Browse button ... and define a custom scope using the Scopes dialog box. Tip Use a special language to define a scope.
Include Test Sources	Select this check box to perform analysis on the test sources.

Specify Inspection Scope Dialog

Code | Inspect Code

Use this dialog box to define the scope to apply inspection to and the profile against which the source code should be inspected.

Note that the list of scopes varies depending on the project type.

Item	Description
Whole project	Select this option to perform the analysis for the whole project.
File <name>	Select this option to analyze the file that is currently selected in the Project tool window or opened in the editor.
Selected files	Select this option to analyze the files that are currently selected in the Project tool window.
Uncommitted files	This scope is only available for the projects under version control. Select this option to have PhpStorm analyze only files that have not been committed to the version control system. Use the drop-down list to further limit the analysis scope. The available options are: − All - select this option to have files from all changelists analyzed. − Default - select this option to have PhpStorm analyze only files from the Default changelist.
Custom scope	Select this option to use a custom scope. Select a pre-defined scope from the drop-down list, or click ... and define the scope in the Scopes dialog . Tip Use a special language to define a scope.
Include test sources	Select this check box to perform analysis on the test sources.
Inspection profile	Select a profile to inspect the specified scope against. A profile is selected from the drop-down list. If the desired profile is not in the list, click the ellipsis button and configure the desired profile on the Inspections page of the Settings dialog.

Structural Search and Replace Dialogs

Edit | Find | Search Structurally

Edit | Find | Replace Structurally

Use these dialog boxes to find and replace fragments of code that structurally match the suggested search template .

Tip To learn more about the possible usages, refer to the section Structural Search and Replace .

Item	Description	Available in
Search template	Use this text area to specify the template to be sought for. You can type the template code in the field or click the Copy existing template button to use one of the existing templates.	Both
Replacement template	Use this text area to specify the template to be substituted. You can type the template code in the field or click the Copy existing template button to use one of the existing templates.	Structural Replace
Save template	Click this button to open the Save Template dialog box, where you have to specify the name of the new template. Note that the new template is stored under the User Defined node of the existing templates tree view.	Both
Edit variables	Calls the Edit variables dialog box to set constraints for template variables.	Both
History	Click this button to open the History dialog box, that shows up to 25 last invoked templates.	Both
Copy existing template	Click this button to open the Existing Templates dialog box, where you can select one of the pre-defined or custom templates. Selected template is displayed in the Preview field. Clicking OK in the Existing Templates dialog box inserts the source code of the template into the Search template or Replace template field.	Both
Recursive matching	If this check box is selected, the search is performed recursively in the results.	Structural Search
Case sensitive	If this check box is selected, the search discerns lower and upper case letters.	Both
File type	Select file type from the drop-down list.	Both
Shorten fully qualified names	This option makes sense in case the template text contains fully qualified class names. If the check box is selected, PhpStorm automatically reduces these names in the template. Otherwise, fully qualified class names are used.	Structural Replace
Reformat according to style	Check this option, if you want PhpStorm to automatically reformat the expanded code fragment according to your code style settings (for details, refer to the Code Style dialog box). If the option is not checked, PhpStorm will only indent the whole template according to the position in code at which it is expanded, leaving its formatting as is.	Structural Replace
Use static import if possible	Check this option, if you want PhpStorm to shorten any references to static elements in the replaced code. PhpStorm will insert a static import for those elements. The elements are then referenced by their short name. If there are no references to static elements in the replaced code, the option will be ignored.	Structural Replace
Scope	Select one of the existing scopes from the drop-down list or click the ellipsis button (alternatively, press `Shift+Enter`), and create your own scope in the Scopes dialog box.	Both
Open in new tab	If this check box is selected, the results of the new search display in a new tab in the Find results tool window. Otherwise, the search results update the existing tab.	Both

Structural Search and Replace. Edit Variable Dialog

Edit | Find | Search Structurally | Edit variables

Use this dialog to define constraints for the variables of a search template .

Tip The contents of the dialog box depend on the selected variable type.

Item	Description
Variables	This area shows a list of variables used in the current search template.
Text constraints	In this area define the following constraints of the selected variable regarding text: – Text/regular expression - in this text box, type a perl-like expression or a class name to be used as a variable constraint. Basic code completion is available for class names. – Invert condition - select this check box to have the text pattern inverted. – Apply constraint within type hierarchy - select this check box to have the search according to the pattern performed both in type names and in parents (within the hierarchy). – Whole words only - when this check box is selected, only whole words within text are matched. This option recognizes string literals and comments.
Occurrences count	In this area, define how pattern hits will be counted. – Minimum count - in this text box, type the minimum number of elements in the list. – Maximum count - in this text box, type the maximum number of elements in the list. – Unlimited - select this check box to allow unlimited number of elements in the list.
Expression constraints	In this area, define how expressions should be processed. – Value is read - if this check box is selected, the matching variable is to be read. – Value is written - if this check box is selected, the matching variable is to be written. – Expression type (regexp) - if the calculated variable is an expression, this constraint checks its type. For instance, for the `foo($a$)` expression the type of the method parameter would be checked. – Expected type of expression (regexp) - if the calculated variable is matched to any expected type of an expression, this constraint checks the expression type anywhere the expression was used. For instance, correspondence between the method parameter type (e.g. `$a$`) in method calls will be checked for methods like `foo($a$)` . – Apply constraint within type hierarchy - select this check box to have the search according to the pattern performed both in type names and in parents (within the hierarchy). – Invert condition - select this check box to have the value of the corresponding check box changed to the opposite one.
Script constraints	In this area, define a variable constraint via a script. Specify the script in the text box or click the ... button to open the Edit Groovy Script Constraint dialog box. The constraint is applied after the initial matching process is finished.
This variable is target of the search	If this check box is selected, the search results will show not the entire expression but the selected variable(s) only.

Edit Variables. Complete Match Dialog

Use the Complete Match dialog to define constraints for the entire pattern that you have specified in the search template .

Item	Description
Text constraints	In this area define the constraints of the selected variable regarding text. In case of Complete Match this area might not be useful. The constraints are as follows: – Text/regular expression - in this text box, type a perl-like expression or a class name to be used as a variable constraint. Basic code completion is available for class names. – Invert condition - select this check box to have the text pattern inverted. – Apply constraint within type hierarchy - select this check box to have the search according to the pattern performed both in type names and in parents (within the hierarchy). – Whole words only - when this check box is selected, only whole words within text are matched. This option recognizes string literals and comments.
Contained in constraints	In this area, define additional constraint pattern inside the already defined search template. Specify the pattern in the text box or click the ... button to open the Existing Templates dialog box. Invert condition - select this check box to have the value of the corresponding field changed to the opposite one.
Script constraints	In this area, define a variable constraint via a script. Specify the script in the text box or click the ... button to open the Edit Groovy Script Constraint dialog box. The constraint is applied after the initial matching process is finished.
This variable is target of the search	If this check box is selected, the search results will show not the entire expression but the selected variable(s) only.

Web Server Debug Validation Dialog

Run | Web Server Debug Validation

Use this dialog box to initiate validation of the debugger setup in a local or remote environment. In the context of PhpStorm, the term remote denotes any environment outside the project root. This can be a server on physically remote host, or a Vagrant box, or a server on the same machine but in a folder outside your project root.

Item	Description
Local Web Server or Shared Folder	Choose this option to check a debugger associated with a local Web server. – Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. For Web servers of the type Inplace, the folder is under the project root. The folder must be accessible through http. – URL to Validation Script: In this text box, type the URL address of the folder where the validation script will be created. If the project root is mapped to a folder accessible through `http`, you can specify the project root or any other folder under it.
Remote Web Server	Choose this option to check a debugger associated with a remote server. – Path to Create Validation Script: In this field, specify the absolute path to the folder under the server document root where the validation script will be created. The folder must be accessible through http. – Deployment Server: In this field, specify the server access configuration of the type Local Server or Remote Server to access the target environment. For details see Configuring Synchronization with a Web Server . Choose a configuration from the drop-down list or click the Browse button ... in the Deployment dialog .
Validate	Click this button to have PhpStorm create a validation script, deploy it to the target remote environment, and run it there.

Settings / Preferences Dialog

File | Settings for Windows and Linux

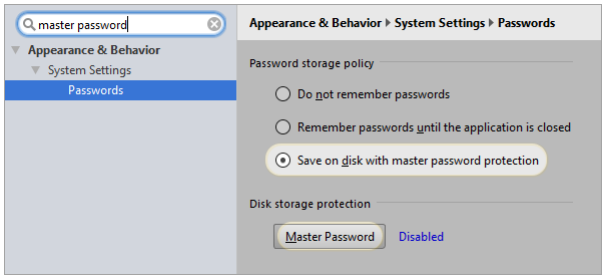
PhpStorm | Preferences for OS X

`Ctrl+Alt+S`

Note The settings that pertain to the current project, are marked with the icon.

The Settings dialog lets you control every aspect of the PhpStorm behavior and appearance.

Use the search box in the upper-left part of the dialog to find the options of interest. Alternatively, you can browse the settings using the hierarchical list of categories (groups of settings) underneath the search box.



On this page find the descriptions of the main controls of the dialog.

Item	Description
Search	Enter a search keyword in the text area. While typing the search string, the list of options in the dialog reduces to the matching occurrences.
	Click this button to clear the search area.
OK	Apply changes and close the dialog box.
Cancel	Discard changes and close the dialog box.
Apply	Apply changes and leave the dialog box opened.
Help	Show reference page.

- Appearance and Behavior
- Keymap
- Editor
- Plugins
- Version Control
- Directories
- Build, Execution, Deployment
- Languages and Frameworks
- Tools
- Framework Integration

Appearance and Behavior

File | Settings | Appearance and Behavior for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior for OS X `Ctrl+Alt+S`

When you select the Appearance and Behavior category in the left-hand pane, its main subcategories are listed in the right-hand part of the dialog.

- Appearance
- Menus and Toolbars
- System Settings
- Notifications
- File Colors
- Scopes
- Quick Lists
- Path Variables

Appearance

File | Settings | Appearance and Behavior | Appearance for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | Appearance for OS X

`Ctrl+Alt+S`

Use this page to change the overall look and feel of your IDE.

- UI Options
- Antialiasing
- Window Options
- Presentation Mode

UI Options

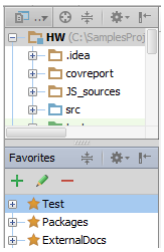
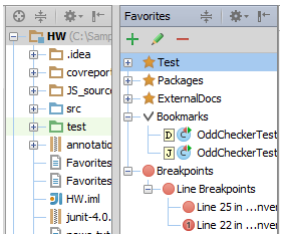
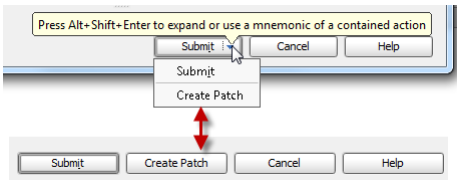
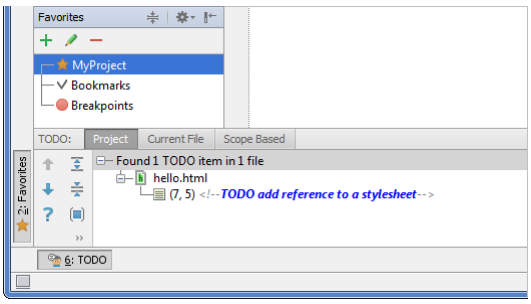
Option	Description
Theme	Use this drop-down list to select the desired theme from the list. In particular, Darcula theme is available. Changing to/from this theme requires PhpStorm restart.
Adjust colors for red-green vision deficiency	Select this option to adjust the IDE colors (code highlighting in the editor, text notifications, etc.) for people with the red-green color deficiency.
Override default fonts by (not recommended)	Select this check box to enable specifying font family and size to be used instead of the default one. When first installed, PhpStorm takes Windows default font size and style.
Cyclic scrolling in list	Select this check box to enable scrolling through a list by jumping from the last item to the first one and vice versa.
Show icons in quick navigation	Select this check box to have icons shown in the quick navigation pop-up menu (`Ctrl` / `Ctrl+Shift` / `Ctrl+Shift+Alt+N`) .
Automatically position mouse cursor on default button	Select this check box to have the mouse pointer placed at the default button when a dialog box opens. If the check box is not selected, the pointer location does not change.
Hide navigation popups on focus loss	If this check box is selected, the navigation pop-up frames (go to class/file/symbol) close, when any other PhpStorm component gets the focus. If this check box is not selected, the navigation pop-up frames persist on changing the focus, and the only way to close such pop-up lays with pressing `Escape` .
Drag-n-Drop with ALT pressed only	If this check box is not selected (by default), PhpStorm allows moving editor tabs, tool window buttons, files and folders in the Project tool window , using drag-n-drop. Select this check box to avoid accidental moving of a file or folder, or a UI component. Thus drag-n-drop only works while ALT key is pressed.
Tooltip initial delay (ms)	Use this slider to specify the time to pass between the moment you hover the mouse over an item in the editor and the moment when the tooltip with its value appears. This settings is especially important during debugging. If the delay is too short using the mouse becomes inconvenient because every mouse move across the screen brings forward a number of tooltips with the values of all the variables.

Antialiasing

Option	Description
IDE	From this drop-down list, select which antialising mode you want to apply to the IDE (including menus, tool windows, etc.): – Subpixel: this option is recommended for LCD displays and takes advantage of the fact that each pixel on a colour LCD is composed of red, green and blue sub-pixels. This allows smoothing text and rendering it with greater detail. – Greyscale: this option is recommended for non-LCD displays or displays positioned vertically. It deals with text at the pixel level. – No antialiasing: this option can be used for displays with high resolution, where non-antialiased fonts are rendered faster and may look better.
Editor	From this drop-down list, select which antialiasing mode you want to apply to the Editor : – Subpixel: this option is recommended for LCD displays and takes advantage of the fact that each pixel on a colour LCD is composed of red, green and blue sub-pixels. This allows smoothing text and rendering it with greater detail. – Greyscale: this option is recommended for non-LCD displays or displays positioned vertically. It deals with text at the pixel level. – No antialiasing: this option can be used for displays with high resolution, where non-antialiased fonts are rendered faster and may look better.

Window Options

Option	Description
Animate windows	Select this check box to have undocked tool windows slide with the animation effect. This option applies only when a tool window is undocked.
Show memory indicator	Select this check box to show the Memory Indicator on the Status Bar .
Disable mnemonics in menu	Select this check box to hide underlining of hot keys in the PhpStorm menus.
Disable mnemonics in controls	Select this check box to hide underlining of hot keys in the PhpStorm controls.
Display icons in menu items	If this check box is selected (by default), the icons are displayed to the left of the menu commands. If this check box is not selected, the menu commands are displayed without icons.

Side by side layout on the left/right	When these check boxes are selected, the way the tool windows are positioned is optimized for a wide-screen display. Side-by-side layout is OFF:  Side by side layout is ON:  Toggle layout by `Ctrl+MouseClick` on splitter between the tool windows. This only applies to the tool windows located on the left and right sides, but not at the top and bottom of the PhpStorm window.
Show tool window bars	Select this check box to display tool window bars.
Show tool window numbers	Select this check box to show tool window quick access numbers on the tool window buttons. You can use the `Alt+number` shortcuts regardless of this setting and change the shortcuts on the Keymap page . Note that the tool window mnemonics show up only when the corresponding keybindings have the format `Alt+n` , where `n` is an integer number in the range from 1 to 9. In the case of a different keyboard shortcut, the mnemonics are not displayed.
Allow merging buttons on dialogs	If this check box is selected, the multiple commands in a dialog box are grouped under a single button with a down arrow. You can view all merged commands by clicking the drop-down list, or pressing `Shift+Alt+Enter` . If this check box is not selected, the buttons will be shown in a row. Compare: 
Small labels in editor tabs	If this check box is selected, the font size on the editor tabs is set to the smaller value. If this check box is not selected, the font size on the editor tabs is set to the default value, as in the project tree view.
Widescreen tool window layout	If this check box is selected, the way the tool windows are positioned is optimized for a wide-screen display. Widescreen tool window layout is OFF:  Favorites MyProject Bookmarks Breakpoints TODO: Project Current File Scope Based Found 1 TODO item in 1 file hello.html (7, 5) <!--TODO add reference to a stylesheet-->>>>

Menus and Toolbars

File | Settings | Appearance and Behavior | Appearance | Menus and Toolbars for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | Appearance | Menus and Toolbars for OS X

`Ctrl+Alt+S`

Use this page to configure the PhpStorm menus and toolbars .

In this topic:

– Menus and Items List

– Controls

Menus and Items List

The list shows the items for the menus and toolbars. The items are grouped according to the areas of their use.

To configure an item, expand the corresponding node and select the desired item. After that, the buttons in the right-hand part of the page become available.

Controls

Item	Description
Add After	Click this button to add a new action to the menu after the selected one. In the Choose Actions to Add dialog box that opens choose the desired action and optionally assign an icon to it.
Add Separator	Click this button to have a separator added to the menu after the selected item.
Edit Action Icon	Click this button to associate an icon with the selected menu item. In the Choose Action Icon Path dialog box that opens specify the path to the desired image. Tip – The image file should have `.png` extension. – The size of the toolbar icons should be 16x16.
Remove	Click this button to delete the selected item from the list.
Move Up	Click this button to move the selected item one position up.
Move Down	Click this button to move the selected item one position down.
Restore All Defaults	Click this button to abandon all the changes made to the all items and return to the default settings.
Restore Default	Click this button to abandon all the changes made to the selected item and return to the default settings.

Choose Actions to Add Dialog

The dialog box opens when you select an item in the Menus and Items List and click the Add After button.

In the dialog box, choose the desired action to be added to the menu or toolbar and optionally assign an icon to it.

Item	Description
Action List	The list shows all the actions available in PhpStorm. The actions are grouped below nodes according to the areas of their use.
Icon Path	In this text box, specify the location of the file with the icon you want to assign to the selected action. If necessary, use the Browse button ... to select the file in the corresponding dialog . Tip – The image file should have `.png` extension. – The size of the toolbar icons should be 16x16.
Set Icon	Click this button to associate the selected action with the icon specified in the Icon Path dialog box.

System Settings

File | Settings | Appearance and Behavior | System Settings

PhpStorm | Preferences | Appearance and Behavior | System Settings

Use this page to configure general behavior of PhpStorm.

Startup/Shutdown

Item	Description
Reopen last project on startup	Select this check box to have PhpStorm re-open the last opened project on startup.
Confirm application exit	Select this check box to have a warning message displayed when you attempt to close PhpStorm.

Project opening

Item	Description
Open project in a new window	Click this radio button to always open a new project in a new window.
Open project in the same window	Click this radio button to always close the current project, and reuse the same window.
Confirm window to open project in	Click this radio button to have PhpStorm ask you whether you want to open a new project in the same frame, or in a new one.

Synchronization

Item	Description
Synchronize files on frame or editor tab activation	If this check box is selected, all the files that were changed externally are reloaded from disk when you switch to PhpStorm from a different application, or when you you switch to their editor tab.
Save files on frame deactivation	If this check box is selected, all modified files are auto saved when you switch from PhpStorm to a different application. Note that you cannot disable autosave completely by turning off this and the following option. See Saving and Reverting Changes .
Save files automatically if application is idle for N seconds	If this check box is selected, all modified files are auto saved at regular time internals. See also, Saving and Reverting Changes .
Use "safe write" (save changes to a temporary file first)	If this check box is selected, a changed file is first saved in a temporary file. If the save operation succeeds, the file being saved is replaced with the saved file. (Technically, the original file is deleted and the temporary file is renamed.) Also, the ownership of such file changes. If this check box is not selected, the ownership of a file does not change, but all the advantages of safe write will be lost.

HTTP Proxy

File | Settings | Appearance and Behavior | System Settings | HTTP Proxy for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | System Settings | HTTP Proxy for OS X

`Ctrl+Alt+S`

If to access the Internet PhpStorm should use an HTTP proxy, specify the proxy settings on this page.

Item	Description
No proxy	Click this radio button to connect to the Internet without a proxy.
Auto-detect proxy settings	Click this radio button to enable using an auto-configuration URL to configure the web proxy settings. When this option is selected, the following controls become enabled:
Manual proxy configuration	Click this radio button to enable manual proxy configuration. When this option is selected, the following controls become enabled:

Item	Description
Automatic proxy configuration URL	Select this check box to manually specify the location of the proxy settings file, in case PhpStorm does not find it automatically.
Clear passwords	Click this button to clear the passwords to the specified proxy.

Item	Description
HTTP	Click this radio button if you want PhpStorm to use an HTTP proxy when accessing the Internet.
SOCKS	Click this radio button if you want PhpStorm to use the Socket Secure protocol when accessing the Internet.
Host name	Specify the proxy hostname or IP address.
Port number	Specify the proxy port number.
No proxy for	Specify here the patterns for the URLs or IP addresses, for which proxy should not be specified.
Proxy authentication	Select this check box if your proxy requires authentication.
Login	Specify the name of the user on whose behalf PhpStorm will connect to the proxy.
Password	Specify the password associated with the user name (login).
Remember password	Select this check box if you want PhpStorm to remember the password. Otherwise, you will be asked to provide the password every time PhpStorm connects to the proxy.

Passwords

File | Settings | Appearance and Behavior | System Settings | Passwords for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | System Settings | Passwords for OS X

Ctrl+Alt+S

Specify whether PhpStorm should remember or save your passwords - ones you use to access password-protected resources such as version control repositories.

If necessary, set a master password. If you do so, instead of the individual passwords - if they are stored in the PhpStorm password database - you'll be able to use your master password.

Note that the password storage policy defined on this page doesn't affect your database passwords. Whether those passwords should be saved is specified for each of the passwords separately, in the settings for the corresponding data sources. See, for example, DB data source settings .

- Password storage policy
- Disk storage protection

Password storage policy

Select one of the following options:

- Do not remember passwords. PhpStorm will ask you for a password each time it needs to access the corresponding password-protected resource.
- Remember passwords until the application is closed. PhpStorm will ask you for a password only the first time it needs to use that password. Then, until you quit the IDE, you won't be asked for that password again.
- Save on disk with master password protection. The passwords will be stored encrypted in the PhpStorm password database. To access this database, a master password will be used.
  Instead of the passwords themselves, PhpStorm will ask you for the master password. For each of the passwords stored in the database, this will happen only once during each session.

Disk storage protection

Master Password. Click this button to set or change the master password. (The Setup Master Password or the Change Master Password dialog will open.)

The master password is used to access the PhpStorm password database and also to strengthen the encryption of the stored passwords.

Enabled/Disabled. The text to the right of the Master Password button reflects the state of the master password. Enabled means that the master password is set and is not empty. Disabled means that the password is not set or is empty.

Setup Master Password Dialog

If your master password is not set yet, you can open this dialog by clicking Master Password on the Passwords page . In certain circumstances, this dialog may also open automatically.

Specify the password that you want to be set as your master password. (The master password is used to access the PhpStorm password database where - if so specified - your passwords are stored. Instead of the individual passwords, you'll be able to use your master password.)

You can set the password empty. As a result, PhpStorm won't ask you for the password when accessing the password database. However, the security of your passwords in that case will be lower.

Item	Description
Password	Specify the master password.
Confirm password	Type the master password once more.
Encrypt with OS user credentials	For Windows only: if you select this check box, the master password itself will be stored encrypted in the password database. (Your Windows logon credentials will be used for encryption.) As a result, PhpStorm won't ask you for the master password when accessing the password database.

Enter Master Password Dialog

This dialog opens automatically when PhpStorm needs to access its password database to retrieve the necessary password.

Specify your master password.

If you don't remember the password, you can reset it (click Reset).

Item	Description
Password	Specify the master password.
Reset	If you don't remember the master password, click this button to reset it. (The Reset Master Password dialog will open.)

Change Master Password Dialog

File | Settings | Passwords - Master Password for Windows and Linux

PhpStorm | Preferences | Passwords - Change Master Password for OS X

`Ctrl+Alt+S`

If your master password is already set, you can open this dialog by clicking Master Password on the Passwords page .

Specify your current master password, and then a new master password. (The master password is used to access the PhpStorm password database where - if so specified - your passwords are stored.)

You can set the new password empty. As a result, PhpStorm won't ask you for the password when accessing the password database. However, the security of your passwords in that case will be lower.

If you don't remember the current password, you can reset the master password (click Reset).

Item	Description
Password	Specify the current master password.
New password	Specify your new master password.
Confirm password	Type the new master password once more.
Encrypt with OS user credentials	For Windows only: if you select this check box, the master password itself will be stored encrypted in the password database. (Your Windows logon credentials will be used for encryption.) As a result, PhpStorm won't ask you for the master password when accessing the password database.
Reset	If you don't remember the current master password, click this button to reset it. (The Reset Master Password dialog will open.)

Reset Master Password Dialog

This dialog opens when you click Reset in the Enter Master Password or the Change Master Password dialog .

Warning! If you reset the master password, all your passwords currently stored in the password database will be lost.

Specify your new master password.

You can set the password empty. As a result, PhpStorm won't ask you for the password when accessing the password database. However, the security of your passwords in that case will be lower.

To come back to the previous dialog, click Enter or Change .

Item	Description
New password	Specify your new master password.
Confirm password	Type the new master password once more.
Encrypt with OS user credentials	For Windows only: if you select this check box, the master password itself will be stored encrypted in the password database. (Your Windows logon credentials will be used for encryption.) As a result, PhpStorm won't ask you for the master password when accessing the password database.
Enter or Change	If you decided not to reset the password (e.g. you remembered it), click this button to come back to the previous dialog. This may be the Enter Master Password or the Change Master Password dialog .

Updates

File | Settings | Appearance and Behavior | System Settings | Updates for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | System Settings | Updates for OS X

`Ctrl+Alt+S`

Use this page to:

- Enable automatic update of PhpStorm and specify to which kind of release you want it updated.
- Obtain information about the current PhpStorm version and availability of a newer version.

Item	Description
Check for updates for	Select this check box to enable the automatic update function, and select the desired update channel (for example, stable version). – Channel Early Access Program: this channel gets patch from the previous EAP/release version. This is not recommended for production development. More details about the Early Access Program, or EAP, are available at http://eap.jetbrains.com/ . – Channel Beta Releases or Public Previews: this channel includes release candidates (RC). – Channel Stable Releases: this channel includes all PhpStorm releases, for example, PhpStorm X.Y.Z Note that the list is only available for the stable versions. For the various EAPs, it is enforced to Early Access Program.
Use secure connection	– If this check box is selected, the secure connection protocol (HTTPS) is used. – If this check box is cleared, the HTTP protocol is used. Note that the HTTP protocol may be blocked due to security reasons. By default, the check box is selected.
Check Now	Click this button to check for updates immediately. Tip You can alternatively choose Help | Check for Updates (for Windows or *NIX) or PhpStorm | Check for Updates (for OS X) on the main menu.
View/edit ignored updates	Follow this link to show/change the builds which were ignored on PhpStorm update. These build numbers are included in the list of ignored updates and not suggested any more.

Usage Statistics

File | Settings | Appearance and Behavior | System Settings | Usage Statistics for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | System Settings | Usage Statistics for OS X

`Ctrl+Alt+S`

Use this page to share the statistics of your PhpStorm usage with JetBrains.

Item	Description
Allow to send usages statistics to JetBrains	Select this check box to allow JetBrains to collect your anonymous statistics.
Daily, Weekly or Monthly	Select one of these options to define how often your usage statistics should be sent to JetBrains.

Notifications

File | Settings | Appearance and Behavior | Notifications for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | Notifications for OS X

`Ctrl+Alt+S`

Use this page to enable and disable notifications about certain events, change their presentation, and optionally enable their logging.

Item	Description
Display balloon notifications	Select this check box to enable event notifications for PhpStorm. (The notifications, generally, are shown in the balloons that appear on the screen when the corresponding events take place.)
Enable system notifications	Select this check box to allow showing system notification. **Warning!** This option is not available on platforms where system notifications are not supported (Windows and some Unixes).
Group	This column lists groups of events that you may be notified of and/or that may be logged.
Popup	If the Display balloon notification check box is selected, the settings in this column specify how the notifications for the corresponding group of events are shown. The available display options are: – Balloon: The balloons with the notification messages appear on the screen for a short period of time and then disappear automatically. The notifications are also shown in the Status bar, and added to the list of notifications. – Sticky balloon: The notification balloons stay on the screen unless you close them. – Tool window balloon: The notification balloons are shown only if an appropriate tool window is open. – No popup: The notifications for the corresponding group of events are not shown.
Log	If the check box for a group of events is selected, the corresponding events are logged and can be seen in the Event Log tool window .

File Colors

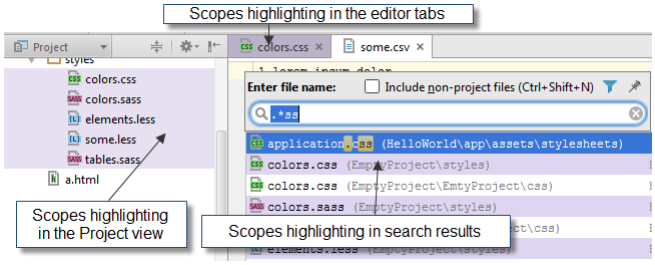
File | Settings | Appearance and Behavior | File Colors for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | File Colors for OS X

`Ctrl+Alt+S`

Use this page to set different background colors for distinguishing between project files and folders of specific scopes . The settings apply to the following UI elements:

- The headers of editor tabs.
- Navigation lists when one searches for files or classes by their names
- Project view



On this page:

- Common Options
- Local Colors
- Shared Colors

Common Options

Item	Description
Enable File Colors	Select this check box to apply background color settings to navigation lists .
Enable Colors in Editor Tabs	Select this check box to apply background color settings to the headers of editor tabs.
Use in Project View	Select this check box to apply background color settings to the Project view.
Manage Scopes	Click this button to open the Scopes dialog in which you can define custom scopes for various actions.

Local Colors

In this area, configure the color-scope associations to be applied locally.

Once defined, a color-scope association cannot be changed. To re-assign a color to a scope, remove the existing association and define a new one.

Item	Tooltip	Description
Scope		This read-only field shows the scope to apply the color setting to.
Color		This read-only field shows the color to be applied to the corresponding scope.
+	Add	Click this button to open the Add Color Label dialog in which you can configure a new color-scope association.
−	Remove	Click this button to remove the selected color-scope association.
or	Move up or Move down	Use these buttons to resort the color-scope associations and thus determine the order in which they are applied.
	Share	Click this button to have the selected scope-color association shared among the members of the team. The selected association will be accordingly moved to the list in the Shared Colors area.

Shared Colors

Use the controls in this area to configure the color-scope associations to be shared among all the members of the team.

Once defined, a color-scope association cannot be changed. To re-assign a color to a scope, remove the existing association and define a new one.

Item	Tooltip	Description
Scope		This read-only field shows the scope to apply the color setting to.
Color		This read-only field shows the color to be applied to the corresponding scope.
+	Add	Click this button to open the Add Color Label dialog in which you can configure a new color-scope association.
−	Remove	Click this button to remove the selected color-scope association.
or	Move up or Move down	Use these buttons to resort the color-scope associations and thus determine the order in which they are applied.
	Unshare	Click this button to have the selected scope-color association applied only locally. The selected association will be accordingly moved to the list in the Local Colors area.

Add / Edit Color Label Dialog

File | Settings | File Colors for Windows and Linux

PhpStorm | Preferences | File Colors for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Add button on the File Colors page. Use this dialog box to configure a new color-scope association.

Item	Description
Scope	From this drop-down list, select the desired scope. The default scopes are: – Non-Project Files – Problems – All – Production – Tests Tip To use custom scopes, click the Manage Scopes button in the File Colors dialog box and define the desired scopes in the Scopes dialog box, that opens.
Color	In this area, select the desired color label to use as the background for displaying the names of files, folders, or packages within the specified scope.

Scopes

File | Settings | Appearance and Behavior | Scopes for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | Scopes for OS X

Ctrl+Alt+S

A scope is a set of files to which various operations apply. Using this dialog, you can define scopes for the various PhpStorm actions, for example, Find Usages , or Code Inspections .

In this section:

- Main toolbar
- Scope configuration controls
- Examples
- Scope toolbar

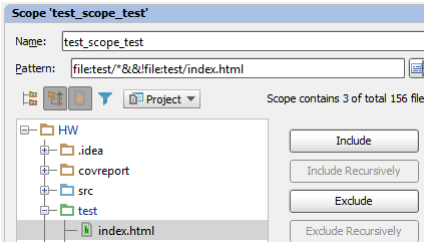
Main toolbar

Item	Tooltip	Description
+	Add scope Insert	Click this button to add a new local or shared scope.
−	Delete Delete	Click this button to delete the selected scope from the list.
	Copy configuration Ctrl+D	Click this button to create a copy of the selected scope.
	Save as	Click this button to have the selected local scope saved as shared or a selected shared scope as local.
	Move Up/Move Down	Use these buttons to move the scopes up and down in the list. If some file is included into several scopes, the order of the scopes becomes important: PhpStorm uses the color of the uppermost scope (shown in the Scopes settings page) to highlight such file. Of course, you can change the order of the scopes, and thus the resulted highlighting. .

Scope configuration controls

Item	Description
Name	In this text box, specify the scope name.
Pattern	In this text box, specify the pattern that defines the current scope. The following elements and structures can be used: 1. The `file:` modifier. The element is mandatory. 2. The `*` asterisk to denote any symbol in a file name or file extension. 3. Logical operators `AND (&&), OR (||)` , and `NOT (!)` . For more information, see Scope Language Syntax Reference . Do one of the following: – Type or edit the pattern manually in the text field Pattern. – Click , or press Shift+Enter to type of edit in the Pattern dialog box. – Choose the desired files in the Project Tree View and use the buttons described below to make PhpStorm generate the corresponding pattern automatically. Warning! Storing empty or incorrect patterns is not allowed. In such cases, you will be prompted with the Syntax Error warning. **Examples** – `file:*.php||file:*.twig` - include all PHP and Twig files. – `file^*php&&!file:*test*.php` - include all PHP files except those that contain the character string `test` in their names. – `file:www/wp-content/themes/my-site-child//*.php||file:www/wp-content/themes/my-site-child//*.css||file:www/wp-content/themes/my-site-child//*.js||file:www/wp-content/plugins/my-site//*.php||file:www/wp-content/plugins/my-site//*.css||file:www/wp-content/plugins/my-site//*.js` - include all .PHP, .CSS, and .JS files from the theme and plugin of the WordPress website. – `file:*.js||file:*.coffee` - include all JavaScript and CoffeeScript files. – `file:*js&&!file:*.min.*` - include all JavaScript files except those that were generated through minification, which is indicated by the `min` extension.
Include	Click this button to have the selected element included in the scope. The corresponding expression is automatically generated and added to the expression in the Pattern text box. Tip If the current element is a folder, the nested subfolders are ignored.
Include Recursively	Click this button to have the selected folder included in the scope, together with the nested subfolders. The corresponding expression is automatically generated and added to the expression in the Pattern text box.
Exclude	Click this button to have the selected element excluded from the scope. The corresponding expression is automatically added to the Pattern. If the current element is a folder, the nested subfolders are ignored.
Exclude Recursively	Click this button to have the selected folder excluded from the scope, together with the nested subfolders. The corresponding expression is automatically added to the Pattern field.

Scope toolbar

ItemTooltipDescription		
Project tree view		The tree view contains all the files available in your project. In the view, select the desired files to be included in the current scope and have the scope definition pattern generated automatically. The message on the toolbar shows the total number of available files and the number of files included in the scope. See also the color legend below. Use the toolbar buttons described below to change the view presentation.
	Show Files	If this button is pressed, source files are displayed explicitly in the tree view. When the files are shown, they can be selected for exclusion/inclusion into a pattern.  If this button is not pressed, the files are hidden.
	Show Included Only	When the button is pressed, the tree shows only the elements that are included in the scope.

Legend of the project tree view

ItemDescription	
Green	Folders and files included in scope.
Black	Folders and files excluded from scope.
Blue	Folders that contain both excluded and included files and subfolders.

Quick Lists

File | Settings | Appearance and Behavior | Quick Lists for Windows and Linux

PhpStorm | Preferences | Appearance and Behavior | Quick Lists for OS X

`Ctrl+Alt+S`

Use this page to configure quick lists. A Quick List is a pop-up menu of PhpStorm commands, configured by the user and associated with a keyboard or mouse shortcut. You can create as many quick lists, as necessary. Each command, included in a quick list, is identified by a sequential number. Numbering starts from the numerals (0 to 9), and then proceeds with the letters in alphabetical order.

Item	Shortcut	Description
+	`Alt+Insert`	Create a new Quick List.
−	`Alt+Delete`	Delete the selected Quick List.
Display name		Edit the name of the selected Quick List.
Description		Edit the description of the selected Quick List. (The description is optional.)
+	`Alt+Insert`	Use this button to add actions to the Quick List. Select the actions in the Add Actions to Quick List dialog that opens.
------		Use this button to add a separator at the end of the Quick List. (Separators help you organize menu commands in logical groups.)
−	`Alt+Delete`	Remove selected actions from the Quick List.
↑	`Alt+U`	Use this button to move the selected item one line up in the list.
↓	`Alt+D`	Use this button to move the selected item one line down in the list.

Path Variables

File | Settings | Appearance & Behavior | Path Variables for Windows and Linux

PhpStorm | Preferences | Appearance & Behavior | Path Variables for OS X

Ctrl+Alt+S

On this page:

- Path Variables page . Use this page to configure path variables and the list of ignored variables .
- Add / Edit Variable dialog . Use this dialog to specify the name and value for a new or existing path variable.
- Example .

Path Variables page

Item	Description
Reset	Use this link to revert path variables and the list of ignored variables to their initial saved states.
Name	This field shows the name of a path variable (readonly).
Value	This field shows the value of a path variable (readonly).
+ (Alt+Insert)	Use this icon or shortcut to create a new path variable. (The Add Variable dialog will open.)
(Enter)	Use this icon or shortcut to edit the selected path variable. (The Edit Variable dialog will open.)
– (Alt+Delete)	Use this icon or shortcut to delete the selected path variable.
Ignored Variables	List the names of the variables that should be ignored . Use semicolons (`;`) to separate individual list items.

Add / Edit Variable dialog

Item	Description
Name	Specify the path variable name.
Value	Specify the path variable value. Use ... (Shift+Enter) to select the necessary file or folder in the Select Path dialog .

Example

Consider storing a library on your disk. This library is attached to your project, and the path to this library is included in the `*.iml` file of your project. However, this path should not be absolute, since the other teammates may store same library in the different locations.

That's why it makes sense to create a dedicated path variable `PATH_TO_LIB` :

1. On the Settings dialog, click Path Variables .
2. Click +.
3. In the Add Variable dialog box, type the variable name `PATH_TO_LIB` , and its value that points to the library location on your disk.
4. Share the `*.iml` file on the version control.
5. The other developers should update their projects, and change the value of `PATH_TO_LIB` variable to point to the locations of their libraries.

Keymap

File | Settings | Keymap for Windows and Linux PhpStorm | Preferences | Keymap for OS X

Use this page to create, edit, and remove custom keymaps for specific environments, and change shortcuts associated with actions.

Note that default keymaps are not editable. To re-configure shortcut associations, create a child keymap based on the desired default one and edit it as required.

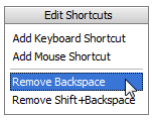
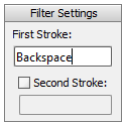
On the other hand, as soon as you try to change a keyboard shortcut associated with an action in one of the default keymaps, a copy of the corresponding keymap is automatically created.

- Keymap Management Buttons
- Keymap Toolbar
- Actions

Keymap Management Buttons

Item	Description
Keymaps	From this drop-down list, select the desired keymap.
Copy	Click this button to create a child keymap on the basis of the keymap selected in the Keymaps drop-down list.
Reset	Click this button to abandon all the changes made to a custom keymap and restore the configuration of the parent keymap.
Delete	Click this button to remove the selected custom keymap from the list.
<language> layout support	This check box appears when a non-English keyboard layout has been detected. When you type any character in the editor, the keyboard layout is recognized and a notification appears. **Warning!** This option is available for Windows and Linux only. If you don't see this option, it means either you are using OS X or PhpStorm didn't detect any of the supported input locales.
Based on keymap	This read-only field shows the name of the parent keymap.

Keymap Toolbar

Item	Tooltip and shortcut	Description
	Expand All `Ctrl+NumPad Plus`	Click this button to expand all nodes in the content pane of actions.
	Collapse All `Ctrl+NumPad -`	Click this button to collapse all nodes in the content pane of actions.
	Edit Shortcut `Enter`	Click this button to change shortcuts for the selected action. It is possible to remove existing shortcuts, and add new ones. Choose the desired change from the drop-down menu:  – Select the option Add Keyboard Shortcut to open the Enter Keyboard Shortcut dialog box, where you can specify the combination of keystrokes to be assigned to the selected action in the current keymap. – Select the option Add Mouse Shortcut to open the Enter Mouse Shortcut dialog box, where you can specify the combination of mouse clicks and buttons to be assigned to the selected action in the current keymap. – Select the options Remove <shortcut> to delete the selected shortcut from the selected action. These commands are duplicated on the context menus of the actions in the Actions content pane.
		Use this text box to search through the content pane of actions. As you type a search string, the actions that match the search pattern are displayed. The previously used search patterns are stored in the search history list. To add the search string to the history list, press `Enter`. Click to reveal the history list of the previous searches. Click to clear the current search pattern from the text box.
	Find Action by Shortcut	Click this button to open the Filter Settings dialog box for filtering out the desired actions by keystrokes. Refer to the section Configuring Keyboard Shortcuts to learn how to specify keyboard shortcuts.  The actions with shortcuts that match the specified criteria are shown in the content pane of actions.
	Clear Filtering	Click this button to restore the initial list of actions in the content pane.

Actions

Item	Description
All Actions	This content pane shows all actions currently available in PhpStorm. The actions are grouped below nodes according to the areas of their use.

- Note that default keymaps are not editable. As soon as you try to change a keyboard shortcut associated with an action, a copy of the corresponding keymap is automatically created.
- If some of the actions have no mapped keyboard shortcuts, they still can be invoked by Go to Action .

Shortcuts	This read-only field shows the list of shortcuts associated with the selected action in the current keymap. Tip The shortcuts are represented depending on the platform. However, some keys are missing on certain keyboard layouts. For example, PageUp / PageDown keys are not available on notebooks. That's why one should use Fn plus arrow keys.
Context menu of an action	
Add Keyboard Shortcut	Choose this command on the context menu of an action to open the Enter Keyboard Shortcut dialog box, where you can specify the combination of keystrokes to be assigned to the selected action in the current keymap.
Add Mouse Shortcut	Choose this command on the context menu of an action to open the Enter Mouse Shortcut dialog box, where you can specify the combination of mouse clicks and buttons to be assigned to the selected action in the current keymap.
Add abbreviation	Choose this command to add an abbreviation that can be used in Search Everywhere .
Remove <shortcut>/<abbreviation>	Choose this command on the context menu of an action to delete the selected shortcut or abbreviation.

Enter Keyboard Shortcut Dialog

The dialog box opens when you select an action and click the Add Keyboard Shortcut button. Use this dialog box to bind the selected action with a new keyboard shortcut, which may consist of one or two keystrokes. The resulting keyboard shortcut is marked with the icon in the Shortcuts list.

Warning! Use your mouse pointer to click buttons in the dialog box. Any keystroke is interpreted as a shortcut!

Item	Description
First Stroke	Use this text box to define the primary shortcut by pressing keyboard keys and key combinations.
Enable	Select this check box to allow an optional second shortcut.
Second Stroke	Use this text box to define an optional shortcut by pressing keyboard keys and key combinations. This field is available after the Enable check box is selected.
Shortcut Preview	In this read-only field, view the newly defined shortcuts.
Conflicts	This read-only field shows messages about conflicts that arise if a suggested keystroke is already in use. **Warning!** You can ignore a conflict and assign a shortcut to several actions. However it is strongly recommended that you avoid binding two actions with the same shortcut, because the priority of these actions is not defined.

Enter Mouse Shortcut Dialog

The dialog box opens when you select an action and click the Add Mouse Shortcut button. Use this dialog box to bind the selected action with a new mouse shortcut, which may be a single or a double clicking one of the mouse buttons or the wheel button.

The resulting mouse shortcut is marked with the icon in the Shortcuts list.

Item	Description
Click Count	In this area, specify the type of mouse click to be assigned to the selected action. The available options are: – Single Click – Double Click
Click Pad	Click the desired mouse button anywhere in this area. **Tip** The number of clicks in this area does not affect the shortcut configuration. No matter how many times you click a button, the click type chosen in the Click Count area will be assigned.
Shortcut Preview	This read-only field shows the newly defined shortcut.
Conflicts	This read-only field shows messages about conflicts that arise if a suggested mouse shortcut is already in use. **Warning!** You can ignore a conflict and assign a shortcut to several actions. However it is strongly recommended that you avoid binding two actions with the same shortcut, because the priority of these actions is not defined.

Editor

File | Settings | Editor for Windows and Linux

PhpStorm | Preferences | Editor for OS X

`Ctrl+Alt+S`

When you select the Editor category in the left-hand pane, its main subcategories are listed in the right-hand part of the dialog.

- General
- Colors and Fonts
- Code Style
- Inspections
- File and Code Templates
- File Encodings
- Live Templates
- File Types
- Copyright
- Emmet
- Images
- Intentions
- Language Injections
- Spelling
- TextMate Bundles
- TODO

General

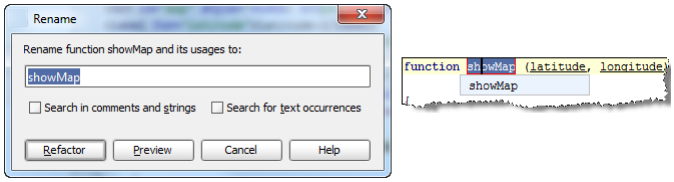
File | Settings | Editor | General for Windows and Linux

PhpStorm | Preferences | Editor | General for OS X

Ctrl+Alt+S

Use the General page of the Settings dialog to configure the editor behaviour and customize its view.

Item	Description
Mouse	
Honor "CamelHumps" word settings when selecting using double click	Select this check box to have PhpStorm invoke the CamelHumps selection when words are selected by double-clicking. This feature works only if the Use 'CamelHumps' words option is enabled.
Change font size (Zoom) with Ctrl/Command+Mouse Wheel	If this check box is selected, a particular editor font size can be changed by rolling the mouse wheel while holding the Ctrl/⌘ key. This check box also affects font size in quick documentation lookup . If this option is unchecked, rolling the mouse wheel while holding the Ctrl/⌘ key scrolls the editor.
Enable Drag'n'Drop functionality in editor	If this check box is selected, you can drag-n-drop code fragments in the editor. Refer to Using Drag-and-Drop in the Editor .
Soft Wraps	
Use soft wraps in editor	If this check box is selected, soft wraps (or word wraps) are used in the editor. The horizontal scroll bar is not normally shown when this option is enabled. However, in certain cases, when a line cannot be "soft-wrapped", the horizontal scroll bar still appears (for example, if a line consists of a single string that is wider than the visible area.)
Use original line's indent for wrapped parts	Select this check box to use custom indentation for soft wraps on resizing the editor or console. Specify the indent value in the Additional shift text field on the right.
Show soft wrap indicators for current line only	If this check box is selected, the soft wrap characters ↵ ↵ will be shown in the active logical line only. Otherwise, soft wraps characters will be shown at the end of each line, and at the beginning of each next line.
Virtual Space	
Allow placement of caret after end of line	If this check box is cleared, the caret never rests after the last symbol in a line.
Allow placement of caret inside tabs	Select this check box to allow placing the caret inside tab characters. The reason is that each tab character shows in the editor as a set of 'virtual' space characters.
Show virtual space at file bottom	If this check box is selected, the currently edited line (even if it is the final line) can be scrolled to the top of the screen. PhpStorm adds the necessary amount of virtual lines.
Other	
Strip trailing spaces on Save	From this drop-down list, select the mode in which PhpStorm will handle trailing spaces in the end of lines on file saving: – Modified lines - Strips trailing spaces only in the end of modified lines. – All - Strips trailing spaces in all lines. – None - Does not strip trailing spaces.
Ensure line feed at file end on Save	Select this check box to have PhpStorm automatically add an empty line in the end of a file during the save procedure.
Show quick documentation on mouse move	Select this check box to show quick documentation for the symbol at caret. The quick documentation pop-up window appears after the specified delay.
Highlight modified lines in gutter	Select this check box if you want added/modified lines to be highlighted with a color stripe in the left editor gutter.
Different color for lines with whitespace-only modifications	This option only becomes available if the Highlight modified lines in gutter option is enabled. Select this check box if you want lines where only whitespaces were added/removed to be highlighted with a different color from lines with more significant modifications.
Highlight on Caret Movement	
Highlight matched brace	Select this check box to have PhpStorm highlight pairs of opening/closing braces when you position the caret right before the opening or right after the closing one. It also works for HTML and XML tags.
Highlight current scope	Select this check box to have PhpStorm highlight the available scope for the code typed in the current caret location.
Highlight usages of element at caret	Select this check box to have PhpStorm highlight all usages of the element at which the caret is currently positioned.
Formatting	
Show notification after reformat code action	Select this check box to show a notification with changes in your code and a shortcut to the Reformat Code dialog every time you try to reformat the code. Otherwise, PhpStorm will reformat code silently.
Show notification after optimize imports action	Select this check box to show notification with changes in your code. Otherwise, PhpStorm will optimize imports silently.
Scrolling	
Smooth scrolling	Select this check box to enable smooth scrolling in the editor.
Prefer scrolling editor canvas to keep caret line centered	Click this option to choose scrolling editor canvas and keeping the caret in place. Keeping the caret in place and scrolling the editor canvas can be helpful in course of debugging session . As you step through the lines of code, the

	editor canvas scrolls, while the line at caret is always in the center of the screen.
Prefer moving caret line to minimize editor scrolling	Click this option to choose moving the caret. When you step through the lines of code during the debugging session , the caret moves down, and the editor canvas doesn't scroll until the caret line reaches the bottom of the screen.
Refactorings	
Enable in-place mode	Select or clear this check box to enable or disable in-place refactorings for PHP. The in-place in connection with the refactorings means specifying all or most of the information necessary for the refactoring by typing, right in the editor. All the affected code fragments are highlighted and change as you type. If appropriate, additional refactoring options are selected in corresponding option boxes. The in-place refactoring mode is available for the following refactorings: – Extract Constant – Extract Field – Extract Variable – Rename If this check box is not selected, the refactoring settings for all of the refactorings are specified in the corresponding dialogs.
Preselect old name	If this check box is selected, the old name of a symbol is selected when the Rename refactoring is invoked for that symbol.  If check box is not selected, the symbol being renamed is not selected.
Show inline dialog for local variables	Select this check box if you want to display a confirmation dialog for the "Inline local variable" refactoring.
Limits	
Maximum number of contents to keep in clipboard	In this text box, specify how many code blocks can be kept in clipboard.
Recent file limit	In this text box, specify how many file names can be included in the list of recent files.
Rich-text copy	
Copy as rich text by default	Select this check box to copy a rich text from the editor to any other editor that recognizes RTF. Note that you can override this option if you select Copy as Plain Text from the context menu in your editor and vice versa, using the Copy as Rich Text option from the context menu overrides the unselected check box in the editor settings.
Color scheme	Use this drop-down list to select a color scheme for the text copy. You can select from the following options: – Default – Active scheme – Darcula
Error highlighting	
Error stripe mark min height (pixels)	In this text box, specify the minimum size of the error and warning stripes.
Autoreparse delay (ms)	In this text box, specify the time period after which PhpStorm starts reparsing the entered text.
'Next Error' action goes to high priority problems only	Select this check box to have PhpStorm pass through the highest priority problems only (for example, errors), when executing Navigate | Next/Previous Highlighted Error command (F2 / Shift+F2). Clear this check box to have PhpStorm pass through all the existing problems (for example, errors and warnings) sequentially.

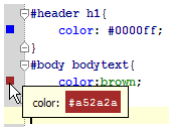
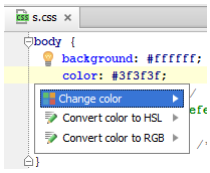
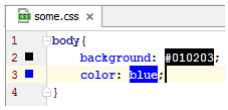
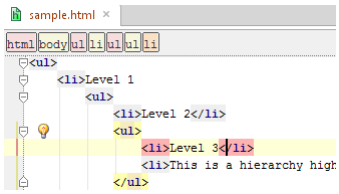
Appearance

File | Settings | Editor | General | Appearance for Windows and Linux

PhpStorm | Preferences | Editor | General | Appearance for OS X

Ctrl+Alt+S

Use this page to customize the appearance of the Editor.

Item	Description
Caret blinking (ms)	Select this check box to make the caret blink with the specified period (in milliseconds).
Use block caret	Select this check box to have the block caret applied in the Insert mode and the usual caret applied in the Overwrite mode. Clear this check box to have the usual caret applied in the Insert mode and the block caret applied in the Overwrite mode.
Show right margin (configured in Code Style options)	Select this check box to have a thin vertical line at the right margin of the editor displayed. Refer to the description of the Code Style settings .
Show line numbers	Select this check box to have line numbering shown in the left gutter area.
Show method separators	Select this check box to have thin lines displayed in PHP and JavaScript classes to separate functions from field declarations.
Show whitespaces	Select this check box to have PhpStorm display white spaces or tabs (depending on the Code Style settings). You can select the following options: – Leader - select this check box to add white spaces before your code line. – Inner - select this check box to display white spaces inside the line of your code. – Trailing - select this check box to display white spaces after the code line.
Show vertical indent guides	Select this check box to have PhpStorm display vertical lines in the editor to indicate positions of indents and thus facilitate typing, manual formatting, reading, and maintaining code.
Show code lens on scrollbar hover	Select this check box to enable lens mode .
Show breadcrumbs	Select this check box to show a breadcrumb trail on top of the editor tab for an HTML or an XML file. Reopen the editor for the changes to take effect. (XML) (HTML)
Show CSS color preview icon in gutter	Select this check box to show color preview icons for the color values. See Changing Color Values in Style Sheets .  If this check box is not selected, it is still possible to invoke the color picker and change color values, by choosing the Change color intention action. 
Show CSS color preview as background	If this check box is selected, the background of the color value shows the color preview: 
Enable XML/HTML tag tree highlighting	Select this check box to show the hierarchy of tags highlighted with different colors. If this option is enabled, you can define the following options: – Levels to highlight: specify the depth of hierarchy to be highlighted. – Opacity: specify brightness of highlighting  Highlighting is activated when there is more than one tag with the same name in the hierarchy.

Auto Import

File | Settings | Editor | General | Auto Import for Windows and Linux

PhpStorm | Preferences | Editor | General | Auto Import for OS X

`Ctrl+Alt+S`

– XML

– TypeScript

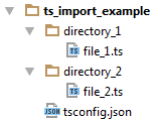
– PHP

XML

Show import pop-up	Automatically display an import pop-up dialog box when typing the name of an unbound namespace.

TypeScript

Show import pop-up	Automatically display import pop-up dialog box when typing the name of a symbol that lacks import statement.
Merge imports for symbols from the same module	

Use paths relative to `tsconfig.json`	When this check box is selected, PhpStorm calculates import paths using the `tsconfig.json` file as the root. When this check box is cleared, PhpStorm calculates import paths relative to the project root.
	For example, if your project is structured as follows:
	 
	
	With the check box selected, PhpStorm generates the following import statement:
	`import {ClassName} from 'directory_2/file_2'`
	If the check box is cleared, the following import statement is generated:
	`import {ClassName} from '../directory_2/file_2'`

PHP

Enable auto-import in file scope	Select this check box to have PhpStorm automatically import PHP namespaces, add use statements, and complete short class names on the fly, when you are typing in a class or file that does not belong to any specific namespace. By default, the check box is cleared.
Enable auto-import in namespace scope	Select this check box to have PhpStorm automatically import PHP namespaces, add use statements, and complete short class names on the fly when you are typing in a class or file that belongs to a certain namespace. The check box is selected by default.

Code Folding

File | Settings | Editor | General | Code Folding for Windows and Linux

PhpStorm | Preferences | Editor | General | Code Folding for OS X

`Ctrl+Alt+S`

Use this page to specify your code folding preferences.

Item	Description
Show code folding outline	Select this check box if you want the code folding toggles (, and) to be shown in the editor. Clear the check box to hide the toggles.
Collapse by default	Select the code fragments which should be folded by default, that is, when a file is first opened in the editor.

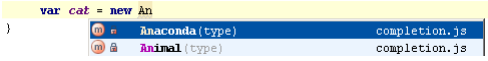
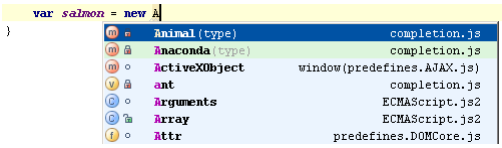
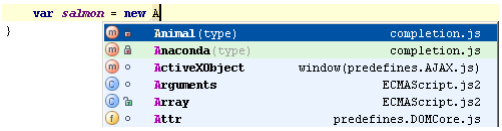
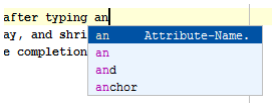
Code Completion

File | Settings | Editor | General | Code Completion for Windows and Linux

PhpStorm | Preferences | Editor | General | Code Completion for OS X

`Ctrl+Alt+S`

Use this page to configure the code completion , and parameter information settings.

ItemDescription	
Code Completion	
Case sensitive completion	From this drop-down list, select the degree to which you want PhpStorm to take into consideration the case sensitivity when suggesting matches for code completion. The available options are: – All: The lookup list includes only those items that match the case of all typed letters. This option is most restrictive.  – None: The lookup list includes all matches regardless of their case.  – First letter: The lookup list includes only the items with the first letter matching. 
Auto-insert when only one choice on:	When the check boxes in this section are selected, PhpStorm doesn't show a lookup list for the corresponding completion type in cases when only one variant of code completion is available, and completes code automatically.
Sort lookup items lexicographically	If this check box is selected, the entries in the suggestion list will be sorted according to their lexical order. If this check box is not selected, the entries in the suggestion list will be sorted by relevance. Note that the check box defines the default behavior. You can change it any time by clicking the A or π icons in the suggestion list. Refer to the section Using Suggestion List for details.
Autopopup code completion	Select this check box, if you want suggestion list to appear after typing anything.  If the check box is not selected, PhpStorm will not suggest code completion automatically.
Insert selected variant by typing dot, space, etc.	If this check box is selected, code is completed by pressing certain character: comma, colon, semicolon, opening parentheses of the various kinds, equality sign, asterisk. This option is turned off by default.
Autopopup documentation (ms) For explicitly invoked completion	Select this check box to have PhpStorm automatically show a pop-up window with the documentation for the class, method, or field currently highlighted in the lookup list. In the text field to the right, specify the delay (in milliseconds), after which the pop-up window should appear. If this check box is not selected, use `Ctrl+Q` to show quick documentation for the element at caret. Quick documentation window will automatically pop up with the specified delay in those cases only, when code completion has been invoked explicitly. For the automatic code completion list, documentation window will only show up on pressing `Ctrl+Q`
Parameter Info	
Autopopup in (ms)	Select this check box to have PhpStorm automatically show a pop-up window with all available method signatures, when an opening bracket is typed in the editor, or a method is selected from the lookup list. In the text field to the right, specify the delay (in milliseconds) after when the pop-up window should appear. If this check box is not selected, use `Ctrl+P` to show the parameter info.
Show full signatures	If this check box is selected, the parameter info displays full signatures, including the method name and returned type.

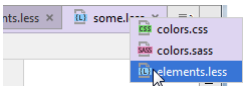
Editor Tabs

File | Settings | Editor | General | Editor Tabs for Windows and Linux

PhpStorm | Preferences | Editor | General | Editor Tabs for OS X

`Ctrl+Alt+S`

Use this page to configure the appearance of the editor tabs and tab headers, specify their positioning on the screen, and define the tab closing policy.

Item	Description
Tab Appearance	
Placement	Use this drop-down list to define the location of the editor tab. The available options are: – Top - the default setting. – Bottom – Right – Left – None - select this option to have single editor without any tabs displayed. Refer to the section Changing Placement of the Editor Tab Headers for details.
Show tabs in single row	Select this check box to have headers of currently opened editor tabs displayed in a single row. As a result, some tab headers may become invisible. To cope with this problem, use the command Window | Editor Tabs | Show All Tabs. Refer to the section Navigating Between Editor Tabs . If this check box is selected, the sorting in alphabetical order mode becomes available for the top and bottom placement of the editor tabs. If this check box is not selected, headers of all the currently opened tabs are displayed, possibly, in several rows.
Hide tabs if there is no space	If this check box is selected, PhpStorm shows as many tabs as fits into the current PhpStorm frame; the rest of the tabs are hidden under the drop-down:  If this check box is not selected, all the editor tabs are shown; so doing, each tab's size reduces: This check box becomes enabled when Show tabs in single row check box is selected.
Hide file extensions in editor tabs	Select this check box to have only file names displayed in editor tab headers.
Show directory in editor tabs for non-unique file names	If this check box is selected, the editor tabs will show the file name together with the parent directory name; if this check box is not selected, only the file name will be included in the editor tab.
Show "close" button on editor tabs	Select this check box to have the Close Active Editor button displayed in editor tab headers.
Mark modified tabs with asterisk	If this check box is selected, changed but yet unsaved files have an asterisk on their editor tabs.
Show tabs tooltips	If this check box is selected, a tooltip with the complete path to a file displays on hovering the mouse pointer over a tab. If this check box is not selected, a tooltip is not shown.
Tab Closing Policy	
Tab limit	In this text box, specify the maximum number of the editor tabs to display.
Navigation from non-modified tab will reuse it	Use this option to specify the PhpStorm behaviour on `Ctrl+Click` (see section Viewing Definition for details). This option allows you to avoid cluttering of the editor space. If this check box is selected, then, if a file in an editor tab has not been modified and the users has navigated from this file, the target file opens in the same tab. If a file has been modified, then the target file opens in a new tab. If this check box is not selected, the target file always opens in a new tab. Note that a file is considered modified if its VCS status has changed .
When number of opened editors exceeds tab limit	In this area, specify which editor tab should be closed when the tab limit is reached and the user attempts to open a new file. The available options are: – Close non-modified files first - if this option is selected, PhpStorm examines the tabs in the order they were opened and closes the first tab with content that has not been modified. – Close less frequently used files - if this option is selected, PhpStorm closes the tab with the less frequently modified content.
When closing active editor	In this area, specify which editor tab to activate when closing the currently active tab. The available options are: – Activate left neighbouring tab - if this option is selected, PhpStorm activates the closest tab to the left from the tab being closed. – Activate right neighbouring tab - if this option is selected, PhpStorm activates the closest tab to the right from the tab being closed. – Activate most recently opened tab - if this option is selected, PhpStorm activates the tab with the file which was opened last.

Gutter Icons

File | Settings | Editor | General | Gutter Icons for Windows and Linux

PhpStorm | Preferences | Editor | General | Gutter Icons for OS X

Ctrl+Alt+S

Use this page to hide or show the icons in the gutter area that invoke actions related to the basic, PhpStorm-wide features or to framework- and technology-specific features for all the newly created editors.

The right-hand pane shows all the gutter icons available in PhpStorm. The basic, PhpStorm-wide features, such as, (Run), are displayed at the top of the list under the Common title. Other features are grouped by the frameworks and technologies to which they are related .

Note that a group of technology-related features is displayed only if the corresponding plugin is installed and activated, see Enabling and Disabling Plugins and Installing, Updating and Uninstalling Repository Plugins for details.

- To have an icon displayed in the gutter area, find the icon or the corresponding action in the list and then select the check box next to it.
- To have an icon hidden, clear the check box next to it.

Postfix Completion

File | Settings | Editor | General - Postfix Completion for Windows and Linux

PhpStorm | Preferences | Editor | General - Postfix Completion for OS X

`Ctrl+Alt+S`

- Overview
- Controls

Overview

Postfix code completion lets you transform an already typed expression to another one based on the postfix you type after a dot, the type of the expression, and its context. This transformation is performed by expanding the postfix-specific predefined template. For example, the `.if` postfix applied to a boolean expression wraps it with an `if` statement. See more at: Postfix Code Completion .

On this page, enable and disable postfix templates and appoint the key to activate the template expansion.

Controls

Item	Description
Enable postfix completion	– Select this check box to have PhpStorm transform expressions with postfixes into other expressions by expanding postfix-specific templates. When the check box is selected, choose the postfixes to apply transformations to by selecting the check boxes next to the desired postfixes in the list below. – When this check box is cleared, no template expansion is applied.
Expand template with	From this drop-down box, choose the key that will invoke template expansion. The available options are: Tab, Space, and Enter.
Table of available postfix templates	The table below shows the list of available postfix templates. To enable or disable a template, select or clear the check box next to it. When you select a template, the right-hand pane shows its description and illustrates how it works by displaying the expression before and after the selected template is applied.

Smart Keys

File | Settings | Editor | General | Smart Keys for Windows and Linux

PhpStorm | Preferences | Editor | General | Smart Keys for OS X

Ctrl+Alt+S

Use this page to enable or disable specific smart keys and to define which actions you want to be invoked automatically.

Item	Description
Home	When this check box is selected, on pressing Home, the caret is positioned at the first non-space character of the current line. Pressing Home subsequently moves the caret from the *Smart Home position* to the first column and back.
End (on blank line)	When this check box is selected, on pressing End in an empty line, the caret is positioned with the indent, which PhpStorm assumes to be reasonable in the current code point (indentation is based on the current Code Style Settings).
Insert pair bracket	Select this check box to have PhpStorm automatically add a closing round or square bracket for each typed opening round or square bracket, respectively.
Insert pair quote	Select this check box to have PhpStorm automatically add a closing single or double quote for each typed opening single or double quote, respectively.
Reformat block on typing "}"	If this check box is selected, then, on typing the closing curly brace, the enclosed code block is reformatted automatically, if the formatting of this code block does not match the selected code style.
Use 'CamelHumps' words	Select this check box to have PhpStorm discern separate words within CamelHump names. Words within a name should start with a capital letter or an underscore. This option impacts some editor actions, for example: – Caret Move (Ctrl+Right / Ctrl+Left) – Caret Move with Selection (Ctrl+Shift+Right / Ctrl+Shift+Left) – Select Word at Caret (Ctrl+W) – Delete to Word Start/End (Ctrl+Backspace and Ctrl+Delete respectively) – Double-clicking
Surround selection on typing quote or brace	If this check box is selected, the selected text on typing a quote, double-quote or brace, will be surrounded with these characters. If this check box is not selected, then the typed quotes, double-quotes or braces will replace the selection.
Add multiple carets on double Ctrl / ⌥ with arrow keys	If this check box is selected, then: – pressing Ctrl (for Windows or *NIX) or ⌥ (for Mac OS) twice plus up/down arrow keys leads to creating multiple carets. – pressing Ctrl (for Windows or *NIX) or ⌥ (for Mac OS) twice plus left/right arrow keys or Home/End leads to creating a selection.
Enter	Use this area to define the actions to be invoked by pressing Enter . – Smart Indent - select this check box to have PhpStorm add a new line and position the caret in it, with the indent that PhpStorm assumes to be reasonable in the current point of code (indentation is based on the current Code Style Settings). If the check box is cleared, upon pressing Enter in a blank line, PhpStorm adds a new line and positions the caret at the current non-space character column. – Insert pair '}' - select this check box to have PhpStorm automatically position a closing brace `}` at the proper column when Enter is pressed in an empty line. In this case PhpStorm seeks backward for the nearest unclosed opening brace `{` and places the closing one at the corresponding indentation level. – Insert documentation comment stub - this check box defines the behavior on pressing Enter after an opening documentation tag. – If this check box is selected, PhpStorm generates a documentation comment stub. For the function comments, this stub contains the required tags (`@param` tags for each parameter declared in the signature, and `@return`). Refer to Creating PHP Documentation Comments , Creating JSDoc Comments for details. – If this check box is not selected, only the closing tag is generated. Warning! Note that this check box refers to JavaScript, and the other languages that have special beginning of documentation comments.
Backspace	Use this drop-down list to define the actions to be invoked by pressing Backspace key. The available options are: – Disabled - pressing Backspace returns the caret by one position at a time. – To nearest indent position – To proper indentation
Reformat on paste	Use this drop-down list to specify how to place pasted code blocks. The available options are: – None - The pasted code is inserted at the caret location as plain text without any reformatting or indenting. – Indent Block - The pasted code block is positioned at the proper indentation level, according to the current Code Style Settings , but its inner structure is not changed. – Indent Each Line- Each line of the pasted code block is positioned at the proper indentation level, according to the current Code Style Settings. – Reformat Block- The pasted code block is reformatted according to the current Code Style Settings. Tip This feature is applicable to lines that contain the trailing line feed characters.
XML/HTML	In this area, define the actions to be invoked automatically when editing XML or HTML code. – Insert closing tag on tag completion: select this check box to have PhpStorm automatically insert a closing XML or HTML tag upon entering the corresponding opening one. – Insert required attributes on tag completion: select this check box to have PhpStorm display a template with all mandatory attributes of the typed tag. – Insert required subtags on tag completion: select this check box to have PhpStorm display a template with all mandatory subtags. – Start attribute on tag completion: select this check box to have PhpStorm display a template with the first mandatory attribute of the typed tag. – Add quotes for attribute value on typing '=': select this check box to have PhpStorm automatically add quotes for the value of the attribute that you are currently typing. – Auto-close tag on typing '</': select this check box to automatically add a closing tag after entering </. Clear this check box to turn off such auto-completion. – Simultaneous <tag></tag> editing: – When this check box is selected and you edit an opening tag the corresponding closing tag is automatically changed accordingly. – If this check box is cleared, editing the opening tag does not affect the closing tag which remains unchanged. As a result, the opening and closing tags

do not match and the entire construct is underlined as erroneous.

This check box controls the behaviour of PhpStorm in the following contexts:

- HTML files
- HTML injections within JavaScript code
- HTML with templates Handlebars/Mustache templates
- Handlebars/Mustache template files with the extension `.hbs`
- XML, XHTML files
- DTD files
- JSX files
- HTML injections in PHP files

CSS

In this area, define the selection of CSS identifiers/classes:

- Select whole CSS identifier on double-click: If this check box is selected, double-click on a CSS identifier or class name selects the entire name up to the prefix:

  `.ic-arrow-left`

  If this check box is not selected, double-click on a CSS identifier or class name selects a portion of a name up to the nearest hyphens:

  `.ic-arrow-left`

AngularJS

Use this area to define the behavior of AngularJS:

- Auto-insert white space in the interpolation: If this check box is selected, a white space is automatically inserted between the braces: `{{ }}`.

  If this check box is not selected, the white space is not inserted: `{{}}`.

PHP

Use this area to define the behavior of PHP:

- Enable smart function parameter completion:

Colors and Fonts

File | Settings | Editor | Colors and Fonts for Windows and Linux

PhpStorm | Preferences | Editor | Colors and Fonts for OS X

Ctrl+Alt+S

On this page:

- Scheme
- Scheme settings

Scheme

Use this section to select the Colors and Fonts scheme. PhpStorm suggests several pre-defined schemes, one of them being the default.

Warning! The selected scheme applies to the editor only!
- The pre-defined schemes cannot be edited. To change colors and fonts settings in such a scheme, create its copy. Refer to the section Configuring Colors and Fonts for details.

Item	Description
Scheme name	From this drop-down list, select the Colors & Fonts scheme to use in your workspace. When a non-default scheme is selected, the node name changes its color to blue. Warning! The pre-defined schemes cannot be edited. To change colors and fonts settings in such a scheme, create its copy.
Save As	Click this button to save the currently selected Colors & Fonts settings as a new scheme. After saving, the scheme settings become editable.
Delete	Click this button to delete the current scheme.

Scheme settings

Use the corresponding pages to change font type, colors and highlighting for the supported languages , consoles, debugger, version control, differences viewer, file statuses, and scopes. Note that besides color settings, highlighting is determined by your Inspection Profile . For example, if you do not want unused symbols to be highlighted, turn off the Unused Symbol inspection in the profile being used.

Choose specific nested page, and configure as required.

Page	Description
Font	Use this page to define the family of the font to be used in the editor, its size, and line spacing. In the Editor Font area you can configure primary and secondary fonts, their size and line spacing. Warning! The options in the Editor Font section can be modified in the editable schemes only. For the default schemes, all the controls are disabled! Change the following options: – Show only monospaced fonts - By default, this check box is selected and the Primary font list will show only monospaced fonts. To choose among all the fonts available in the system, clear this check box. If an entered character cannot be displayed in the current font, the closest match is used. – Primary font list shows the selection of all the fonts, detected on the computer. – Secondary font list shows the selection of fonts, which will be used if PhpStorm fails to use a primary one. – Enable font ligatures: select this check box to show the typographic ligatures . Note that this feature depends on the selected fonts.
General	Use this page to customize the font type and colors for the editor textual components, specified in the list.
Console Colors	Use this page to define the colors to be used in the run or debug console (see Run Tool Window , Debug Tool Window. Console).
Console Font	Use this page to define the family of the font to be used in the run or debug console, its size, and line spacing (see Run Tool Window , Debug Tool Window. Console). In the Console Font area you can configure primary and secondary fonts, their size and line spacing. Warning! The options in the Console Font section can be modified in the editable schemes only. For the default schemes, all the controls are disabled! Change the following options: – Show only monospaced fonts - By default, this check box is selected and the Primary font list will show only monospaced fonts. To choose among all the fonts available in the system, clear this check box. If an entered character cannot be displayed in the current font, the closest match is used. – Primary font list shows the selection of all the fonts, detected on the computer. – Secondary font list shows the selection of fonts, which will be used if PhpStorm fails to use a primary one. – Enable font ligatures: select this check box to show the typographic ligatures . Note that this feature depends on the selected fonts. – Apply editor font settings: click this button to apply editor font settings to the console fonts.
Custom	Use this page to customize the font type and colors in user-defined file types , in particular, for custom keywords highlighting .
VCS	Use this page to customize the colors of the left gutter markers and VCS annotations . To change color of any VCS component, click the corresponding color swatch to show the Color Picker dialog, and select the desired color.
File Status	Use this page to customize font type and colors to visually denote the current file status with regards to a VCS repository .
By Scope	Use this page to customize colors and highlighting for class members assigned to various scopes. To edit a scope, click the Edit Scopes button below the pane with a list of existing scopes. The Scopes dialog box opens.
Languages and	Use these pages to customize font type and colors for the corresponding file types and the PhpStorm components.

Code Style

File | Settings | Editor | Code Style for Windows and Linux

PhpStorm | Preferences | Editor | Code Style for OS X

`Ctrl+Alt+S`

Use this page to choose the code style scheme and change it as required. Settings of a code style scheme are automatically applied every time PhpStorm generates, refactors, or reformats your code.

On this page:

- Scheme
- Line Separators
- Indents Detection
- Formatter Control
- EditorConfig

Scheme

In this area, choose the code style scheme and change it as required. Settings of a code style scheme are automatically applied every time PhpStorm generates, refactors, or reformats your code.

The code styles are defined at the project level and at the global level.

- At the Project level, settings are grouped under the Project scheme, which is predefined and is marked in bold font. The Project style scheme applies to the current project only.
  The default scheme cannot be changed! When any modifications are made, PhpStorm automatically creates a copy of the default scheme. If you want to use a different project level scheme, choose the desired scheme and copy its settings to project.
- At the Global level, settings are grouped under the predefined Default scheme, and any other scheme created by the user. The global settings are used when the user doesn't want to keep code style settings with the project and share them.

Item	Description
Scheme	From this drop-down list, select the scheme to be used. The predefined schemes are shown in bold. The custom schemes, ones created as copies of the predefined schemes, are in plain text.
Manage	Click this button to open the Code Style Schemes dialog that lets you manage the schemes.
Reset	This link appears in the upper-right corner of the dialog when the current style or scheme are modified. Click this button to discard changes.

Line Separators

PhpStorm lets you configure line separator and indentation options for various languages. When reformatting source code , PhpStorm will apply the specified indentation behavior and skip the sections denoted with the special formatting off/on markers.

Item	Description
Line Separator (for new files)	Use this drop-down list to specify which line separator is to be used in files created by PhpStorm. The available options are: – System dependent - choose this option to use the default selection. – Unix and OS X (\n) - choose this option to use the Unix and OS X line separator. – Windows (\r\n) - choose this option to use the Windows line separator. – Classic Mac (\r) - choose this option to use the Classic Mac line separator. Refer to the section Configuring Line Separators .
Right Margin (columns)	In this text box, specify the number of columns to be used to display pages in the editor.
Wrap when typing reaches right margin	Select this check box to ensure that edited text always fits in the specified right margin.

Indents Detection

Use this area to specify default options for indentation.

Item	Description
Detect and use existing file indents for editing	Select this check box for PhpStorm to detect the existing indents in the file and use them for editing instead of the indents specified in the Code Style settings for the specific language.
Show notifications about detected indents	Select this check box to show a notification if PhpStorm detects indents that are different from the ones specified in the Code Style settings for the specific language.

Formatter Control

In this area, specify the markers to limit code fragments that you want to exclude from reformatting, see more in Reformatting Source Code . In the source code, formatting markers are written inside line comments, see Commenting and Uncommenting Blocks of Code .

Item	Description
Enable formatter markers in comments	– If this check box is selected, the fragments of code between the line comments with the formatting markers will not be reformatted but will preserve the original formatting. After you select the check box, the fields below become available and you can specify the character strings to be treated as formatting markers. – If the check box is cleared, the formatting markers will be ignored and the code between the line comments with markers will be reformatted.
Markers	
Formatter off:	In this text box, specify the character string that will indicate the beginning of a code fragment which you want to exclude from reformatting. Type a character string

	with the `@` symbol in preposition or leave the predefined value `@formatter:off`.
Formatter on:	In this text box, specify the character string that will indicate the end of a code fragment which you want to exclude from reformatting. Type a character string with the `@` symbol in preposition or leave the predefined value `@formatter:on`.
Regular expressions	Select this check box to use regular expressions instead of specifying the formatting markers explicitly. PhpStorm matches formatter on/off markers using the regular expression specified instead of the exact string.

Example of using formatting markers

The original source code

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested))
    {
        return false;
    }
//@formatter:on
require_once $paths['public'] . '/index.php';
```

The code after reformatting

When the formatting markers are disabled, the original formatting is broken:

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested)) {
    return false;
}
//@formatter:on
require_once $paths['public'] . '/index.php';
```

When the formatting markers are enabled, the original formatting is preserved:

```
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = urldecode($uri);
$paths = require __DIR__ . '/bootstrap/paths.php';
$requested = $paths['public'] . $uri;
//@formatter:off
if ($uri !== '/' and file_exists($requested))
    {
        return false;
    }
//@formatter:on
require_once $paths['public'] . '/index.php';
```

EditorConfig

In this area enable the support of the EditorConfig plugin.

Item	Description
Enable EditorConfig support	Select this check box to enable the EditorConfig plugin support. In this case you can specify your own code style settings that override the IDE settings. However, if you decide to use IDE settings after creating the EditorConfig settings file then you need clear the Enable EditConfig support check box. See also Configuring Code Style procedure.

Code Style Schemes

File | Settings | Code Style - Manage for Windows and Linux

PhpStorm | Preferences | Code Style - Manage for OS X

Ctrl+Alt+S

Use this dialog box to manage the set of code style schemes.

Item	Description
Save As...	Click this button to create a copy of the currently selected scheme. This new scheme can be used for copying a scheme to project, and for export .
Delete	Click this button to remove the currently selected scheme. Note that the predefined schemes cannot be deleted.
Copy to Project	Click this button to copy the settings of the currently selected scheme to project. When the settings are copied, PhpStorm suggests to switch to this scheme.
Import	Click this button to import an IntelliJ IDEA XML code style settings, or JSCS config file.
Export	Click this button to export to an IntelliJ IDEA XML code style settings file.
Close	Click this button to close the Code Style Schemes dialog box.

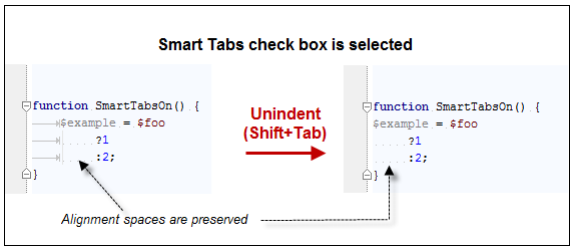
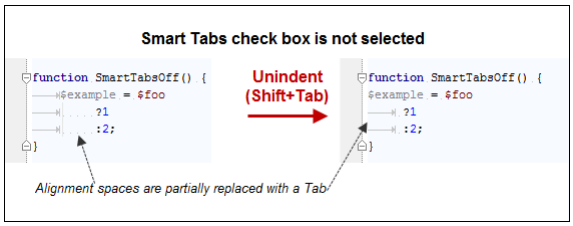
Note that any changes to the set of schemes (creating copies, or deleting unnecessary schemes) only take place on clicking Apply or OK buttons in the Settings/Preferences dialog .

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list.

The results are displayed in the Preview pane.

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces.  – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment.  The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Wrapping and braces

In this tab, customize the code style options, which PhpStorm will apply on reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Right Margin (columns)

Use Right Margin field to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings is used.

Wrap on typing

Use Wrap on typing settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options:

- Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings .
- Yes - in this case PhpStorm uses the value specified in the Right Margin field.
- No - in this case this option is switched off and a line can exceed the value specified in the right margin.

Keep when reformatting

Use the check boxes to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line breaks* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the Line breaks check box before you reformat the source code .

Wrapping options

The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, method call arguments, or assignment statements).

Item	Description
Wrapping style	From this drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. With this option selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.

Alignment options

Item	Description
Align when multiline	If this check box is selected, a code construct starts at the same column on each next line. Otherwise, the position of a code construct is determined by the current indentation level.
Align parenthesised when multiline	If this check box is selected, a parenthesised code construct starts at the column of the opening parenthesis each next line. Otherwise, the position of a code construct is determined by the current indentation level.
`<character(s)>` on next line	Select this check box to have the specified character or characters moved to the next line when the lines are wrapped.
'else' on new line	Use this check box to have the corresponding statements or characters moved to the next line.
New line after `<character>`	Select this check box to have the code after the specified character moved to a new line.
Special else if treatment	If this check box is selected, `else if` statements are located in the same line. Otherwise, `else if` statements are moved to the next line to the corresponding indent level.
Indent case branches	If this check box is selected, the `case` statement is located at the corresponding indent level. Otherwise, `case` statement is placed at the same indent level with `switch` .

Braces placement options

Item	Description
Braces placement style	Use this drop-down list to specify the position of the opening brace in class declarations, method declarations, and other types of declarations. The available options are: – End of line - select this option to have the opening brace placed at the declaration line end. – Next line if wrapped - select this option to have the opening brace placed at the beginning of the line after the multiline declaration line. – Next line - select this option to have the opening brace placed at the beginning of the line after the declaration line. – Next line shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level. – Next line each shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level, and have the next line shifted to the next indent level as well.

Force braces	From this drop-down list, choose the braces introduction method for `if` , `for` , `while` , and `do () while` statements. The available options are:
	– Do not force - select this option to suppress introducing braces automatically.
	– When multiline - select this option to have braces introduced automatically, if a statement occupies more than one line. Note that PhpStorm analyzes the number of lines in the entire statement but not only its condition. For example, if the code is
	`<?php`
	`if (true)`
	`return false;`
	PhpStorm will insert braces automatically:

```php
<?php
                    if (true)
                    {
                    return false;
                    }
```

– Always - select this check box to have braces always introduced automatically.

Code Style. PHP

File | Settings | Editor | Code Style | PHP for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | PHP for OS X

Use this page to configure formatting options for PHP files. View the result in the Preview pane on the right.

On this page:

- Set from...
- PHP-secific formatting settings for Tabs and Indents
- PHP-specific formatting settings for Spaces
- PHP-specific formatting settings for Wrapping and Braces
- Blank lines
- PHPDoc
- Other
- Arrangement

Set from...

Click this link to choose the base for the current language default code style from the pop-up list, that appears. The list contains two options:

- Language: choose this option to inherit the coding style settings from another language. Select the source language from the list, that opens. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.
- Predefined code style: choose this option to use the coding standards defined for a specific framework. Select one of the following frameworks from the list:
  - PEAR
  - Zend
  - Symfony2
  - PSR1 / PSR2
  - WordPress
  - Drupal
  - Joomla!

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

PHP-secific formatting settings for Tabs and Indents

See Common Code Style Options: Tabs and Indents for the settings that are common for all languages.

ItemDescription	
Indent as template language unless PHP-only	– When this check box is selected, PhpStorm applies the indentation style specified for the template language of the source code into which PHP fragments are embedded. If the PHP code is not surrounded with any other code, the PHP style is applied depending on the status of the Use tab character check box. – When this check box is cleared, the PHP style is applied no matter whether the PHP code fragment is embedded or the entire file is `.php`.

PHP-specific formatting settings for Spaces

See Common Code Style Options: Spaces for the settings that are common for all languages.

Select or clear the check boxes to have spaces inserted, not inserted, or removed in the following PHP contexts:

ItemDescription	

Before Parentheses	– Array initializer parentheses
	Select this check box to have a space inserted before the opening parenthesis in array declarations, such as:
	`$array = array (0 => "zero", 1 => "one");`
	If the check box is cleared, no space is inserted and the code looks as follows:
	`$array = array(0 => "zero", 1 => "one");`

Around Operators	– Concatenation Select this check box to have spaces inserted before and after the concatenation operator (`.`) so the code looks as follows: `echo "The result is " . $i;` When the check box is cleared, no spaces are inserted around the concatenation operator so the code looks as follows:

```
echo "The result is ".$i;
```

Before Left Brace	– Class left brace
	Select this check box to have a space inserted between the class name and the opening brace in class declarations:
	
	`class Class1 {`
	`function Foo()`
	`}`
	
	When the check box is cleared, no space is inserted:
	
	`class Class1{`
	`function Foo()`
	`}`
	
	Selecting or clearing the check box is relevant only when Braces placement in class declarations is set to End of line on the Wrapping and Braces tab.

In Ternary Operator (`?:`)	Select the check boxes in this section to have spaces automatically inserted before and after `?`, before and after `:`, and between `?` and `:` in ternary (conditional) operators .
Other	Select the check boxes in this section to have spaces automatically inserted before and after commas, semicolons, unary NOT operators (`!`), and after type casts. – After type cast When this check box is selected, PhpStorm automatically inserts a space after the closing parentheses of a cast : `$fst = (string) $foo;` If the check box is cleared, no spaces is inserted and the casted variable sticks to the cast: `$fst = (string)$foo;`

PHP-specific formatting settings for Wrapping and Braces

See Common Code Style Options: Wrapping and Braces for the settings that are common for all languages.

Item	Description
Braces placement	In this section, choose the position for opening braces in declarations of namespaces, classes, and functions, in loops, and in other constructions. Choose the required position from the drop-down list, the available options are: – End of line - select this option to have the opening brace placed at the declaration line end. – Next line if wrapped - select this option to have the opening brace placed at the beginning of the line after the multiline declaration line. – Next line - select this option to have the opening brace placed at the beginning of the line after the declaration line. – Next line shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level. – Next line each shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level, and have the next line shifted to the next indent level as well.

Extends/implements list	In this section, configure wrapping and subsequent alignment of extend and implements lists:
	– Do not wrap - when this option is selected, no special wrapping style is applied.
	With this option selected, the nested alignment and braces settings are ignored.
	– Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation.
	– Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation.
	– Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.
	– Align when multiline
	When this check box is selected, each item in an extends or an implements list, which after wrapping starts on a new line, are aligned by the first item, which remains unwrapped:
	
	`abstract class Foo`
	`extends`
	`Class1 implements`
	`Class2,`
	`Class8 {`
	`}`
	
	When the check box is cleared, the position of each item in a wrapped list is determined by the chosen indentation level.
	
	The status of the check box affects the formatting only if you have chosen to wrap lists, if the Do not wrap option is chosen, selecting or clearing the check box has no effect.

Extends/implements keyword	In this section, configure wrapping for the extends and implements keywords in class declarations.
	If you choose the Do not wrap option, no wrapping will be applied, if you choose Wrap if long or Wrap always, each keyword and each item in an extends or implements list will be displayed on a new line:
	
	`abstract class Foo`
	`extends`
	`Class1`
	`implements`
	`Class2,`
	`Class8 {`
	`}`

Function declaration parameters

In this area, configure formatting in declarations of functions and methods.

- Keep ')' and '{' on one line
  When this check box is cleared, the opening curly brace is moved to the next line:

```
function Foo()
                    {
                    }
```

When the check box is selected, the opening curly brace is displayed on the same line as the function parameters:

```
function Foo() {
                    }
```

Chained method calls

In this section, configure wrapping and subsequent alignment of chained calls.

- Do not wrap - when this option is selected, no special wrapping style is applied.
  With this option selected, the nested alignment and braces settings are ignored.
- Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation.
- Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation.
- Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.
- Align when multiline
  When the check box is selected, each called method, which after wrapping is positioned on a new line, is aligned by the first one, which remains unwrapped:

```
$x = $x->one( "a", "b" )
                    ->two( "c", "d", "e" )
                    ->three( "fg" )
                    ->four;
```

When the check box is cleared, the position of each item in a wrapped list is determined by the chosen indentation level.

The status of the check box affects the formatting only if you have chosen to wrap lists, if the Do not wrap option is chosen, selecting or clearing the check box has no effect.

- Place ';' on new line
  When the check box is cleared, the semicolon (`;`) is displayed after the last item in a chained call. If the check box is selected, the semicolon is moved to a new line:

```
$x = $x->one( "a", "b" )
                    ->two( "c", "d", "e" )
                    ->three( "fg" )
                    ->four
                    ;
```

Assignment statement

In this section, configure wrapping and subsequent alignment in assignment statements.

- Do not wrap - when this option is selected, no special wrapping style is applied.
  With this option selected, the nested alignment and braces settings are ignored.
- Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation.
- Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation.
- Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.
- Assignment sign on new line
- Align consecutive assignments
  When this check box is selected, the assignment signs in consecutive assignment statements are aligned by the rightmost one:

```
$y            = foo( $x
                    );
                    $intermediate = foo( $intermediate
                    );
                    $result       = $result . $y . $intermediate;
```

When this check box is cleared, no alignment is applied:

```
$y = foo(
                    $x
                    );
                    $intermediate = foo( $intermediate
                    );
                    $result = $result . $y
                    . $intermediate;
```

Class field/constant

In this section, configure wrapping and subsequent alignment within lists of class properties (fields) or class constants .

groups	– Align fields in columns – Align constants

Array initializer	In this section, configure wrapping and subsequent alignment in array declarations.
	– Do not wrap - when this option is selected, no special wrapping style is applied.
	With this option selected, the nested alignment and braces settings are ignored.
	– Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation.
	– Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation.
	– Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.
	– Align when multiline
	When the check box is selected and the New line after '(' check box is cleared, each element, which after wrapping is positioned on a new line, is aligned by the first one, which remains unwrapped:
	`$colours = array( "blue",`
	`"red",`
	`"white",`
	`"green",`
	`"yellow" );`
	When both the Align when multiline and the New line after '(' check boxes are selected, all the elements ae aligned according to the indentation settings:
	`$colours = array(`
	`"blue",`
	`"red",`
	`"white",`
	`"green",`
	`"yellow" );`
	Selecting or clearing the Align when multiline and the New line after '(' check boxes affects the formatting only if you have chosen to wrap lists, if the Do not wrap option is chosen, the status of the check boxes has no effect.
	– New line after '('
	When this check box is selected, the first element of the array is displayed on a new line and all the elements of the array are aligned according to the indentation settings, regardless of the status of the Align when multiline check box.
	When this check box is cleared, the first element of the array remains on the same line, and the other elements are aligned according to the indentation settings.
	– Place ')' on new line

Modifier list	– Wrap after modifier list
	When this check box is selected, the code is wrapped after a list of visibility modifiers:
	`protected`
	`function Foo() {`
	`}`
	When the check box is cleared, no wrapping is performed:
	`protected function Foo() {`
	`}`

Blank lines

Use this tab to define where and how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of location, specify the number of blank lines to be inserted. The results are displayed in the Preview pane.

Item	Description
Keep Maximum Blank Lines	In this area, specify the number of blank lines to be kept after reformatting in the specified locations.
Minimum Blank Lines	In the text boxes in this area, specify the number of blank lines to be present in the specified locations. **Warning!** These settings do not influence the number of blank lines before the first and after the last item.

PHPDoc

In this tab, configure the code style to be applied inside PHPDoc comments. Learn more about documenting PHP code at Creating PHP Documentation Comments .

Item	Description
Align parameter names	Select this check box to have the `&<paramname>` elements aligned.
Keep blank lines	Select this check box to suppress removing blank lines automatically.
Blank lines around parameters	Select this check box to have a blank line inserted above and below the section with `@param` tags.
Blank line before the first tag	Select this check box to have an blank line inserted above the first PHPDoc tag.
Align tag comments	Select this check box to have the `description` elements aligned.
Wrap long lines	Select this check box to have the text that exceeds the right margin wrapped to the next line.
Generated Doc Blocks	In this area, configure the code style to be applied within generated PHP documentation blocks, see Creating PHP Documentation Comments .

	– Use fully-qualified class names: select this check box to have PhpStorm specify fully qualified class names for properties, function parameters, `return` and `throws` values, etc.
`@throws` Tag Analysis	– Call tree analysis depth: specify the depth of analysis. PhpStorm can analyze exceptions up to 3 levels deep. – Ignore Runtime exceptions: If this check box is selected, the exceptions which can only be found at runtime (either based on `RuntimeException` or any of its subclasses), are ignored. – Ignore Logic exceptions: If this check box is selected, exceptions caused by the error in the program logic (either based on `LogicException` or any of its subclasses), are ignored.

Other

Item	Description
Indent code in PHP tags	Select this check box to have the code enclosed in `<?php>` tags indented against the opening `<?php` tag.
Convert True/False constants to upper case	Select this check box to have the `true` and `false` constants displayed in the upper case.
Convert Null constant to upper case	Select this check box to have the `null` constant displayed in the upper case.
Blank line before return statement	Select this check box to have PhpStorm automatically insert a blank line before each `return` statement.

Spaces around variables/expressions in brackets	– Select this check box to have PhpStorm insert spaces inside brackets during reformatting only if the brackets enclose a variable or an expression. This setting affects reformatting only if you have not configured force insertion of spaces inside brackets by selecting the Brackets check box under the Within node in the Spaces tab.
	This option helps you keep your code in accordance with the WordPress PHP Coding Standards .
	– If this check box is cleared, the spaces insertion policy depends on the settings under the Within node in the Spaces tab regardless of the type of content inside brackets:
	  – if the Brackets check box is selected, spaces are always inserted
	  – if the Brackets check box is cleared, spaces are never inserted.

Code Commenting	In this area, configure the code style options to be applied to comments. – Line comment at first column: - select this check box to have line comments start at the first column, without any indentation. Note that no extra blank spaces are added after the line comment characters. The check box is by default selected. When the check box is cleared, the line comments start from the minimum indentation within the selected code block to be commented.
Array declaration style	– Force short declaration style: select this check box to have PhpStorm replace the `array()` constructs with `[]` in array declarations during reformatting. When the check box is cleared, the traditional literal style in array declarations is preserved after reformatting. – Align key-value pairs: select this check box to have the `=>` separators in key-value assignments aligned. – Add a comma after last element in multiline array: select this check box to have PhpStorm automatically insert a comma after the last item in declarations of multiline arrays to meet the required coding standard, for example, the Symfony coding standards . See Arrays. Syntax for details.

Arrangement

In this tab, define a set of rules to re-arrange your PHP code according to your preferences.

Item	Description
Grouping Rules	Use this area to set the grouping rules. – Keep getters and setters together Select this check box to keep getter and setter methods together. By default, this check box is selected. – Keep overridden methods together Select this check box to group the overridden methods together by class and interface. In order: list, select keep or by name options. – Keep dependent methods together Select this check box to group the dependent methods together. In order: list, select depth-first or breadth-first options.
Matching rules	Use this area to define elements order as a list of rules, where every rule has a set of matches such as modifier or type. – + - use this button to add a rule. The empty rule dialog window opens. – — - use this button to remove the rule from the list. – - use this button to edit an existing rule. To see this button, navigate to the rule that you want to edit and click on the button. In pop-up window that opens, modify the rule fields. – - use these buttons to move the selected rule up or down.
Empty rule	Use this window to create a new matching rule or edit an existing one. You can select from the following filters: – Type - use this filter to choose classes or methods for your rule. Note that clicking twice on the type keyword will negate the condition. – Name - use this field to specify entry names in the rule. This filter matches only entry names such as field names, method names, class names and etc. The filter supports regular expressions and uses a standard syntax . The match is performed against the entire name. – Order - use this drop-down list to select the sorting order for the rule. This option is useful when more than one element uses the same matching rule. In this case, selecting Keep order will keep the same order as was set before the rearrangement and selecting Order by Name will sort the elements with the same matching rule by their names. – Aliases - this option displays aliases that were defined in the Rules Alias Definition dialog. You can remove the ones you do not need.
	This icon appears when you select Order by Name in Order option. The icon indicates that the items in this rule are alphabetized.

Code Style. CoffeeScript

File | Settings | Editor | Code Style | CoffeeScript for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | CoffeeScript for OS X

Use this page to configure formatting options for CoffeeScript files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Spaces
- Wrapping and braces
  - Right Margin (columns)
  - Wrap on typing
  - Keep when reformatting
  - Wrapping options
  - Alignment options
  - Braces placement options
- Blank lines
- Other
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list. The results are displayed in the Preview pane.

Wrapping and braces

In this tab, customize the code style options, which PhpStorm will apply on reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Right Margin (columns)

Use Right Margin field to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings is used.

Wrap on typing

Use Wrap on typing settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options:

- Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings .
- Yes - in this case PhpStorm uses the value specified in the Right Margin field.
- No - in this case this option is switched off and a line can exceed the value specified in the right margin.

Keep when reformatting

Use the check boxes to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line breaks* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the Line breaks check box before you reformat the source code .

Wrapping options

The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, method call arguments, or assignment statements).

Item	Description
Wrapping style	From this drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. With this option selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.

Alignment options

Item	Description
Align when multiline	If this check box is selected, a code construct starts at the same column on each next line. Otherwise, the position of a code construct is determined by the current indentation level.
Align parenthesised when multiline	If this check box is selected, a parenthesised code construct starts at the column of the opening parenthesis each next line. Otherwise, the position of a code construct is determined by the current indentation level.
`<character(s)>` on next line	Select this check box to have the specified character or characters moved to the next line when the lines are wrapped.
'else' on new line	Use this check box to have the corresponding statements or characters moved to the next line.
New line after `<character>`	Select this check box to have the code after the specified character moved to a new line.
Special else if treatment	If this check box is selected, `else if` statements are located in the same line. Otherwise, `else if` statements are moved to the next line to the corresponding indent level.
Indent case branches	If this check box is selected, the `case` statement is located at the corresponding indent level. Otherwise, `case` statement is placed at the same indent level with `switch`.

Braces placement options

Item	Description
Braces placement style	Use this drop-down list to specify the position of the opening brace in class declarations, method declarations, and other types of declarations. The available options are: – End of line - select this option to have the opening brace placed at the declaration line end. – Next line if wrapped - select this option to have the opening brace placed at the beginning of the line after the multiline declaration line. – Next line - select this option to have the opening brace placed at the beginning of the line after the declaration line. – Next line shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level. – Next line each shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level, and have the next line shifted to the next indent level as well.

Force braces	From this drop-down list, choose the braces introduction method for `if`, `for`, `while`, and `do () while` statements. The available options are:
	– Do not force - select this option to suppress introducing braces automatically.
	– When multiline - select this option to have braces introduced automatically, if a statement occupies more than one line. Note that PhpStorm analyzes the number of lines in the entire statement but not only its condition. For example, if the code is
	`<?php if (true) return false;`
	PhpStorm will insert braces automatically:
	`<?php if (true) { return false; }`
	– Always - select this check box to have braces always introduced automatically.

Blank lines

Use this tab to define where and how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of location, specify the number of blank lines to be inserted. The results are displayed in the Preview pane.

Item	Description
Keep Maximum Blank Lines	In this area, specify the number of blank lines to be kept after reformatting in the specified locations.
In code	Use this field to set the number of the blank lines.

Other

Item	Description
Align object properties	From the drop-down list, select the type of objects' alignment: – Do not align: the attributes in sequential lines will be not aligned. – On colon: the attributes in sequential lines will be aligned against the colon. – On value: the attributes in sequential lines will be aligned against the value.
Line comments at first column	Select this check box to place a line comment in the first column.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. CSS

File | Settings | Editor | Code Style | CSS for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | CSS for OS X

Use this page to configure formatting options for CSS files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Other
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Other

In this tab, specify the alignment, braces and spaces options to be applied on reformatting.

Item	Description
Braces placement	Use this drop-down list to specify where PhpStorm should place the opening braces of selectors. The available options are: – At the end of line – Next line
Align values	Use this drop-down list to specify how PhpStorm should align attributes and values. The available options are: – Do not align: select this option to specify alignment on the first character of an attribute name. – On value: select this option to specify alignment on the first character of the value of an attribute. – On colon
Blank lines between blocks	In this text box, specify the minimum number of sequential blank lines to be retained after reformatting.
Align closing brace with properties	If this check box is selected, the closing brace of the selector will be placed under the list of properties. If this check box is not selected, the closing brace of the selector will be placed under the selector.
Keep single-line blocks	If this check box is selected, the blocks with a single property will be confined to one line. If this check box is not selected, each property will be placed to its own line.
Spaces	Select the check boxes in this area to add a space after the colon delimiting key and value, and before the opening brace of the selector.
HEX Colors	Use this area to configure the hex color syntax. You can select from the following check options: – Convert hex colors to - select this check box to configure the hex color letter case. You can choose Lower case or Upper case. – Convert hex colors format to - select this check box to configure the hex color format length. You can choose Long format or Short format. View changes in the Preview pane.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style: EJS

File | Settings | Code Style - EJS for Windows and Linux

PhpStorm | Preferences | Code Style - EJS for OS X

`Ctrl+Alt+S`

On this page, configure formatting options to be applied within EJS inclusions in your code. PhpStorm recognizes such inclusions based on these delimiters specified on the EJS page of the Settings dialog box. The default delimiters are `<%` and `%>`. For more details about using EJS, see http://embeddedjs.com/getting_started.html.

View the result in the Preview pane.

Item	Description
Spaces within execute tag	Select this check box to have spaces inserted between the opening (`<%`) and closing (`%>`) delimiters and the template body inside them. Otherwise, the template body is not separated from the delimiters.
Spaces within execute and get escaped tag	Select this check box to have spaces inserted between the template body on the one hand and the opening/closing delimiters or escape tags (`=`) on the other. Otherwise, the template body immediately follows and precedes the delimiters and tags.
Spaces within execute and get unescaped tag	Select this check box to have spaces inserted between the template body on the one hand and the opening/closing delimiters or unescape tags (`==`) on the other. Otherwise, the template body immediately follows and precedes the delimiters and tags.
Spaces within generate tag	Select this check box to have spaces inserted between the template body on the one hand and the opening/closing delimiters or generate tags (`%`) on the other. Otherwise, the template body immediately follows and precedes the delimiters and tags.
Spaces within filter tag	Select this check box to have spaces inserted between the template body on the one hand and the opening/closing delimiters or filter tags (`%=:`) on the other. Otherwise, the template body immediately follows and precedes the tags.
Before ':'	Select this check box to have a space inserted before a colon (`:`).
After ':'	Select this check box to have a space inserted after a colon (`:`).
Chain '|'	Select this check box to have spaces inserted before and after a vertical bar (`|`).

Code Style. Gherkin

File | Settings | Editor | Code Style | Gherkin for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | Gherkin for OS X

Use this page to configure formatting options for Gherkin files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description

Use tab character	– If this check box is selected, tab characters are used:
	  – On pressing the `Tab` key
	  – For indentation
	  – For code reformatting
	– When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. HAML

File | Settings | Editor | Code Style | HAML for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | HAML for OS X

Use this page to configure formatting options for HAML files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. HTML

File | Settings | Editor | Code Style | HTML for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | HTML for OS X

Use this page to configure formatting options for HTML files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Other
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Other

Item	Description
Right Margin	Use these settings to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings will be used.
Wrap on typing	Use these settings settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options: – Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings . – Yes - in this case the value in the specified right margin is used. – No - in this case this option is switched off and a line can exceed the number that is specified in the right margin.
Keep line breaks	Select this check box to have PhpStorm honor line breaks when reviewing HTML files in the editor.
Keep line breaks in text	Select this check box to have PhpStorm honor line breaks in attributes (for example, lengthy descriptions) when reviewing HTML files in the editor.
Keep blank lines	In this text box, specify the minimum number of sequential blank lines to be retained after reformatting.
Wrap attributes	Use this drop-down list to determine how attribute lines should be wrapped. The available options are: – Do not wrap - if this option is selected, no special wrapping style is applied to the code. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped to give one element per line with proper indentation. – Wrap always - select this option to have all elements in lists wrapped to give one element per line with proper indentation.
Wrap text	Select this check box to have long lines wrapped according to the code style settings.
Align attributes	Select this check box to have attributes in sequential lines aligned.
Align text	Select this check box to have PhpStorm align the text that occupies several lines within a tag.
Keep white spaces	Select this check box to suppress replacing actual white spaces with tabs.
Spaces	In this area, define the use of spaces for attributes and tag names. – Around "=" in attribute - select this check box to have spaces added around the "=" symbol in attributes. – After tag name - select this check box to have spaces added after tag names. – In empty tag - select this check box to have spaces added in empty tags.

Insert new line before	This display field shows a list of tags before which a new line should be inserted. Use the button next to the field or press `Shift+Enter` to open the Insert New Line Before Tags dialog box, where you can edit the list of tags.
Remove new line before	This display field shows a list of tags before which a break line should be removed. Use the button next to the field or press `Shift+Enter` to open the Remove Line Breaks Before Tags dialog box, where you can edit the list of tags.
Do not indent children of	This display field shows a list of tags whose children should not be indented. Use the button next to the field or press `Shift+Enter` to open the Do Not Indent Children Of dialog box, where you can edit the list of tags.
Or if tag size more than	In this text box, specify the minimum length of a tag in lines starting from which its children are not indented.
Inline elements	This display field shows a list of tags that are presented in the source code in the same line with the other tags. If a tag is removed from the list, the editor automatically moves it to a new line, when you add such tag to the source code. Use the button next to the field or press `Shift+Enter` to open the Inline Elements dialog box, where you can edit the list of tags.
Keep white spaces inside	This display field shows a list of tags inside which you want the editor to preserve white spaces *as is*, without any changes. Use the button next to the field or press `Shift+Enter` to open the Keep Whitespaces Inside dialog box, where you can edit the list of tags.
Don't break if inline content	This display field shows a list of tags that are not to be wrapped if their content is inlined. Use the button next to the field or press `Shift+Enter` to open the Don't Wrap If Inline Content Only dialog box, where you can edit the list of tags.
Generated quote marks	Choose the style of the quote marks (double, single, or none) to be automatically inserted around HTML attributes on typing `=`. This is important when HTML is inserted dynamically using JavaScript or PHP and you want to consistently use double-quote pairs for JavaScript or PHP strings and single-quote pairs for HTML to prevent problems, for example, when copying and pasting.
Enforce on format	If this check box is selected, then on code reformatting the previously generated quote marks will be replaced (for example, double quotes with single quotes).

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. Pug/Jade

File | Settings | Editor | Code Style | Pug(Jade) for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | Pug(Jade) for OS X

Use this page to configure formatting options for Pug(Jade) files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Before you start working with Pug(Jade), make sure that Pug(ex-Jade) plugin is installed and enabled . The plugin is not bundled with PhpStorm.

Tabs and Indents

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the `Tab` key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. JavaScript

File | Settings | Editor | Code Style | JavaScript for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | JavaScript for OS X

Use this page to configure formatting options for JavaScript files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Spaces
- Wrapping and braces
  - Right Margin (columns)
  - Wrap on typing
  - Keep when reformatting
  - Wrapping options
  - Alignment options
  - Braces placement options
- Blank lines
- Other
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Indent chained methods	In declarations of functions, the second and further methods in a chain are displayed on a separate line. – When the check box is selected, the second and further methods in a chain are aligned with the first call. – When the check box is cleared, the second and further methods in a chain are aligned with the object on which they are invoked.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list. The results are displayed in the Preview pane.

Wrapping and braces

In this tab, customize the code style options, which PhpStorm will apply on reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Right Margin (columns)

Use Right Margin field to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings is used.

Wrap on typing

Use Wrap on typing settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options:

- Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings .
- Yes - in this case PhpStorm uses the value specified in the Right Margin field.
- No - in this case this option is switched off and a line can exceed the value specified in the right margin.

Keep when reformatting

Use the check boxes to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line breaks* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the Line breaks check box before you reformat the source code .

Wrapping options

The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, method call arguments, or assignment statements).

Item	Description
Wrapping style	From this drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. With this option selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.

Alignment options

Item	Description
Align when multiline	If this check box is selected, a code construct starts at the same column on each next line. Otherwise, the position of a code construct is determined by the current indentation level.
Align parenthesised when multiline	If this check box is selected, a parenthesised code construct starts at the column of the opening parenthesis each next line. Otherwise, the position of a code construct is determined by the current indentation level.
`<character(s)>` on next line	Select this check box to have the specified character or characters moved to the next line when the lines are wrapped.
'else' on new line	Use this check box to have the corresponding statements or characters moved to the next line.
New line after `<character>`	Select this check box to have the code after the specified character moved to a new line.
Special else if treatment	If this check box is selected, `else if` statements are located in the same line. Otherwise, `else if` statements are moved to the next line to the corresponding indent level.
Indent case branches	If this check box is selected, the `case` statement is located at the corresponding indent level. Otherwise, `case` statement is placed at the same indent level with `switch`.

Braces placement options

Item	Description
Braces placement style	Use this drop-down list to specify the position of the opening brace in class declarations, method declarations, and other types of declarations. The available options are: – End of line - select this option to have the opening brace placed at the declaration line end. – Next line if wrapped - select this option to have the opening brace placed at the beginning of the line after the multiline declaration line. – Next line - select this option to have the opening brace placed at the beginning of the line after the declaration line. – Next line shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level. – Next line each shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level, and have the next line shifted to the next indent level as well.

Force braces	From this drop-down list, choose the braces introduction method for `if`, `for`, `while`, and `do () while` statements. The available options are:
	– Do not force - select this option to suppress introducing braces automatically.
	– When multiline - select this option to have braces introduced automatically, if a statement occupies more than one line. Note that PhpStorm analyzes the number of lines in the entire statement but not only its condition. For example, if the code is
	`<?php if (true) return false;`
	PhpStorm will insert braces automatically:
	`<?php if (true) { return false; }`
	– Always - select this check box to have braces always introduced automatically.

Blank lines

Use this tab to define where and how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of location, specify the number of blank lines to be inserted. The results are displayed in the Preview pane.

Item	Description
Keep Maximum Blank Lines	In this area, specify the number of blank lines to be kept after reformatting in the specified locations.
In code	Use this field to set the number of the blank lines.

Other

Item	Description
Align object properties	From the drop-down list, select the type of objects' alignment: – Do not align: the attributes in sequential lines will be not aligned. – On colon: the attributes in sequential lines will be aligned against the colon. – On value: the attributes in sequential lines will be aligned against the value.

Use semicolon to terminate statements	Select this check box to have statements terminated with a semicolon.
Generated Code	In this area, configure the code style in `use strict` statements, is you have selected the Prefer Strict Mode check box on the JavaScript page of the Settings dialog box. – String literal type: from this drop-down list, choose the type of quotes to be used in `use strict` statements. The available options are Single quotes and Double quotes.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. JSON

File | Settings | Editor | Code Style | JSON for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | JSON for OS X

Use this page to configure formatting options for JSON files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Spaces
- Blank lines
- Wrapping and braces
  - Right Margin (columns)
  - Wrap on typing
  - Keep when reformatting
  - Wrapping options
  - Alignment options
- Other

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent chained methods	In declarations of functions, the second and further methods in a chain are displayed on a separate line. – When the check box is selected, the second and further methods in a chain are aligned with the first call. – When the check box is cleared, the second and further methods in a chain are aligned with the object on which they are invoked.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list. The results are displayed in the Preview pane.

Blank lines

Use this tab to define where and how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of location, specify the number of blank lines to be inserted. The results are displayed in the Preview pane.

Item	Description
Keep Maximum Blank Lines	In this area, specify the number of blank lines to be kept after reformatting in the specified locations.
In code	Use this field to set the number of the blank lines.

Wrapping and braces

In this tab, customize the code style options, which PhpStorm will apply on reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Right Margin (columns)

Use Right Margin field to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings is used.

Wrap on typing

Use Wrap on typing settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options:

- Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings .
- Yes - in this case PhpStorm uses the value specified in the Right Margin field.
- No - in this case this option is switched off and a line can exceed the value specified in the right margin.

Keep when reformatting

Use the check boxes to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line breaks* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the Line breaks check box before you reformat the source code .

Wrapping options

The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, method call arguments, or assignment statements).

Item	Description
Wrapping style	From this drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. With this option selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.
Ensure right margin is not exceeded	If this check box is selected, the formatter will do its best to avoid having document lines exceeding the right margin. This option takes precedence over the Do not wrap wrapping style.

Alignment options

Item	Description
Align when multiline	If this check box is selected, a code construct starts at the same column on each next line. Otherwise, the position of a code construct is determined by the current indentation level.
Align parenthesised when multiline	If this check box is selected, a parenthesised code construct starts at the column of the opening parenthesis each next line. Otherwise, the position of a code construct is determined by the current indentation level.
`<character(s)>` on next line	Select this check box to have the specified character or characters moved to the next line when the lines are wrapped.
'else' on new line	Use this check box to have the corresponding statements or characters moved to the next line.
New line after `<character>`	Select this check box to have the code after the specified character moved to a new line.
Special else if treatment	If this check box is selected, `else if` statements are located in the same line. Otherwise, `else if` statements are moved to the next line to the corresponding indent level.
Indent case branches	If this check box is selected, the `case` statement is located at the corresponding indent level. Otherwise, `case` statement is placed at the same indent level with `switch` .

Other

Item	Description
Align object properties	From the drop-down list, select the type of objects' alignment: – Do not align: the attributes in sequential lines will be not aligned. – On colon: the attributes in sequential lines will be aligned against the colon. – On value: the attributes in sequential lines will be aligned against the value.

Code Style. Less

File | Settings | Editor | Code Style | Less for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | Less for OS X

Use this page to configure formatting options for Less files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. Sass

File | Settings | Editor | Code Style | Sass for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | Sass for OS X

Use this page to configure formatting options for Sass files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description

Use tab character	– If this check box is selected, tab characters are used:
	  – On pressing the `Tab` key
	  – For indentation
	  – For code reformatting
	– When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. SCSS

File | Settings | Editor | Code Style | SCSS for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | SCSS for OS X

Use this page to configure formatting options for SCSS files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description

Use tab character	– If this check box is selected, tab characters are used:
	  – On pressing the `Tab` key
	  – For indentation
	  – For code reformatting
	– When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. SQL

File | Settings | Editor | Code Style | SQL for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | SQL for OS X

Use this page to configure formatting options for SQL files. View the result in the Preview pane on the right.

On this page:

- General
- Tabs and Indents
- Spaces
- Wrapping and Braces
- Blank Lines
- Set from...

General

Use this tab to specify certain formatting behaviors in your code. The results are displayed in the Preview pane.

Item	Description
Word Case	In this node, specify whether the keywords, identifiers, or quoted identifiers case should change automatically. Click the right-hand column, and check the desired behavior on the menu to turn it on. The possible options are: - To upper: the elements are automatically converted to the upper case. - To lower: The elements are automatically converted to lower case. - Do not change: The case of elements is left as is.
Identifier quotations	Click the right-hand column, and check the desired behavior on the menu to turn it on. The possible options are: - Quote: the identifiers are automatically quoted. - Unquote: The identifiers are automatically unquoted. - Do not change: The identifiers are left as is.
New line before/after	Use these nodes to define how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of SQL element, select of clear the check box to the right. The results are highlighted in the Preview pane.
New line around semicolon	Use this check box to manage line delimiters around a semicolon.
Alignment	Select the check boxes next to the desired elements of source code to have them automatically aligned.

Tabs and Indents

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the `Tab` key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	- If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. - If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list. The results are displayed in the Preview pane.

Wrapping and Braces

In this tab, customize the code style options, which PhpStorm will apply when reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Item	Description
Keep When Reformatting	Use the check boxes in this node to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line brakes* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the *Line brakes* check box before you reformat the source code .
Wrap inside	Use this node to define wrapping style for the lengthy elements. You can use different wrapping style options that are available from the drop-down list in the left-hand pane.
Values expression	Use this option to define wrapping style for values expressions. You can use different wrapping style options that are available from the drop-down list in the left-hand pane.
Wrapping Style Options	The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, expressions, or assignment statements). From the drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. Note If this option is selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation.

Blank Lines

Use this tab to insert blank lines into your code.

Item	Description
Keep Maximum Blank Lines	Use this area to specify blank lines in your code. Use In code field to enter the number of lines you want to insert. The default number is 2.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. Stylus

File | Settings | Editor | Code Style | Stylus for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | Stylus for OS X

Use this page to configure formatting options for Stylus files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the `Tab` key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	- If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. - If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style: Twig

File | Settings | Code Style - Twig for Windows and Linux

PhpStorm | Preferences | Code Style | lang for OS X

`Ctrl+Alt+S`

Use this page to configure formatting options for Twig templates. View the result in the Preview pane.

On this page:

- Tabs and Indents
- Tabs and Indents
- Other
- Set from...

Tabs and Indents

Tabs and Indents

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the `Tab` key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Other

In this tab, configure the coding style for Twig special syntax. PhpStorm automatically applies these settings in accordance with the Symfony2 standards when reviewing files with Twig templates in the editor.

Item	Description
Spaces inside delimiters `{%...%}`	Select this check box to have PhpStorm automatically insert one and only one space after and before each opening (`{%`) and each closing delimiter (`%}`) of a Does something tag.
Spaces inside delimiters `{{...}}`	Select this check box to have PhpStorm automatically insert one and only one space after and before each opening (`{{`) and each closing delimiter (`}}`) of a Says something tag.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. TypeScript

File | Settings | Editor | Code Style | TypeScript for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | TypeScript for OS X

Use this page to configure formatting options for TypeScript files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Spaces
- Wrapping and braces
  - Right Margin (columns)
  - Wrap on typing
  - Keep when reformatting
  - Wrapping options
  - Alignment options
  - Braces placement options
- Blank lines
- Other
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent chained methods	In declarations of functions, the second and further methods in a chain are displayed on a separate line. – When the check box is selected, the second and further methods in a chain are aligned with the first call. – When the check box is cleared, the second and further methods in a chain are aligned with the object on which they are invoked.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Spaces

Use this tab to specify where you want spaces in your code. To have PhpStorm automatically insert a space at a location, select the check box next to this location in the list. The results are displayed in the Preview pane.

Wrapping and braces

In this tab, customize the code style options, which PhpStorm will apply on reformatting the source code . The left-hand pane contains the list of exceptions (Keep when reformatting), and placement and alignment options for the various code constructs (lists, statements, operations, annotations, etc.). The right-hand pane shows preview.

Alignment takes precedence over indentation options .

Right Margin (columns)

Use Right Margin field to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings is used.

Wrap on typing

Use Wrap on typing settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options:

- Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings .
- Yes - in this case PhpStorm uses the value specified in the Right Margin field.
- No - in this case this option is switched off and a line can exceed the value specified in the right margin.

Keep when reformatting

Use the check boxes to configure exceptions that PhpStorm will make when reformatting the source code. For example, by default, the *Line breaks* check box is selected. If your code contains lines that are shorter than a standard convention, you can convert them by disabling the Line breaks check box before you reformat the source code .

Wrapping options

The wrapping style applies to the various code constructs, specified in the left-hand pane (for example, method call arguments, or assignment statements).

Item	Description
Wrapping style	From this drop-down list, select the desired wrapping style: – Do not wrap - when this option is selected, no special wrapping style is applied. With this option selected, the nested alignment and braces settings are ignored. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Wrap always - select this option to have all elements in lists wrapped so that there is one element per line with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped so that there is one element per line with proper indentation.

Alignment options

Item	Description
Align when multiline	If this check box is selected, a code construct starts at the same column on each next line. Otherwise, the position of a code construct is determined by the current indentation level.
Align parenthesised when multiline	If this check box is selected, a parenthesised code construct starts at the column of the opening parenthesis each next line. Otherwise, the position of a code construct is determined by the current indentation level.
`<character(s)>` on next line	Select this check box to have the specified character or characters moved to the next line when the lines are wrapped.
'else' on new line	Use this check box to have the corresponding statements or characters moved to the next line.
New line after `<character>`	Select this check box to have the code after the specified character moved to a new line.
Special else if treatment	If this check box is selected, `else if` statements are located in the same line. Otherwise, `else if` statements are moved to the next line to the corresponding indent level.
Indent case branches	If this check box is selected, the `case` statement is located at the corresponding indent level. Otherwise, `case` statement is placed at the same indent level with `switch` .

Braces placement options

Item	Description
Braces placement style	Use this drop-down list to specify the position of the opening brace in class declarations, method declarations, and other types of declarations. The available options are: – End of line - select this option to have the opening brace placed at the declaration line end. – Next line if wrapped - select this option to have the opening brace placed at the beginning of the line after the multiline declaration line. – Next line - select this option to have the opening brace placed at the beginning of the line after the declaration line. – Next line shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level. – Next line each shifted - select this option to have the opening brace placed at the line after the declaration line being shifted to the corresponding indent level, and have the next line shifted to the next indent level as well.

Force braces	From this drop-down list, choose the braces introduction method for `if` , `for` , `while` , and `do () while` statements. The available options are:
	– Do not force - select this option to suppress introducing braces automatically.
	– When multiline - select this option to have braces introduced automatically, if a statement occupies more than one line. Note that PhpStorm analyzes the number of lines in the entire statement but not only its condition. For example, if the code is
	`<?php if (true) return false;`
	PhpStorm will insert braces automatically:
	`<?php if (true) { return false; }`
	– Always - select this check box to have braces always introduced automatically.

Blank lines

Use this tab to define where and how many blank lines you want PhpStorm to retain and insert in your code after reformatting. For each type of location, specify the number of blank lines to be inserted. The results are displayed in the Preview pane.

Item	Description
Keep Maximum Blank Lines	In this area, specify the number of blank lines to be kept after reformatting in the specified locations.
In code	Use this field to set the number of the blank lines.

Other

Item	Description
Align object properties	From the drop-down list, select the type of objects' alignment:

- Do not align: the attributes in sequential lines will be not aligned.
- On colon: the attributes in sequential lines will be aligned against the colon.
- On value: the attributes in sequential lines will be aligned against the value.

Use semicolon to terminate statements	Select this check box to have statements terminated with a semicolon.
Generated Code	In this area, configure the code style in `import` statements, see Creating Imports , section Importing TypeScript Symbols. – String literal type: from this drop-down list, choose the type of quotes to be used in `import` statements. The available options are Single quotes and Double quotes.
Use 'public' modifier	Use this check box to have the public access modifier inserted or omitted in the generated code. For example, during generation of a public method from the following:

```
class Test {
                    public test():void {
                    var x = 1;
                    }
                    }
```

– If the check box is selected, the public access modifier is automatically inserted in the generated code:

```
class Test {
                    public test():void {
                    this.extracted();
                    }
                    extracted(){
                    var x = 1;
                    }
                    }
```

– If the check box is cleared, the public access modifier is omitted during code generation:

```
class Test {
                    public test():void {
                    this.extracted();
                    }
                    public extracted(){
                    var x = 1;
                    }
                    }
```

See TypeScript Language Handbook , chapter Private/Public Modifiers .

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. XML

File | Settings | Editor | Code Style | XML for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | XML for OS X

Use this page to configure formatting options for XML files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Other
- Arrangement
- Set from...

Tabs and Indents

Item	Description
Use tab character	– If this check box is selected, tab characters are used: – On pressing the `Tab` key – For indentation – For code reformatting – When the check box is cleared, PhpStorm uses spaces instead of tabs.
Smart tabs	– If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces. – If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment. The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Continuation indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted between the elements of an array, in expressions, method declarations and method calls.
Keep indents on empty lines	If this check box is cleared, then PhpStorm will keep indents on the empty lines as if they contained some code. If this check box is not selected, PhpStorm will delete the tabulation characters.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Other

Item	Description
Right Margin	Use these settings to specify a margin space required on the right side of an element. If you select Default option then a value of the right margin from the global settings will be used.
Wrap on typing	Use these settings settings to specify how the edited text is fitted in the specified Right margin. You can select one the following options: – Default - in this case PhpStorm uses the Wrap on typing option that is specified in the global settings . – Yes - in this case the value in the specified right margin is used. – No - in this case this option is switched off and a line can exceed the number that is specified in the right margin.
Keep line breaks	Select this check box to have PhpStorm honor line breaks when reviewing XML files in the editor.
Keep line breaks in text	Select this check box to have PhpStorm honor line breaks in attributes (for example, lengthy descriptions) when reviewing XML files in the editor.
Keep blank lines	In this text box, specify the minimum number of sequential blank lines to be retained after reformatting.
Wrap attributes	Use this drop-down list to determine how attribute lines should be wrapped. The available options are: – Do not wrap - if this option is selected, no special wrapping style is applied to the code. – Wrap if long - select this option to have lines going beyond the right margin wrapped with proper indentation. – Chop down if long - select this option to have elements in lists that go beyond the right margin wrapped to give one element per line with proper indentation. – Wrap always - select this option to have all elements in lists wrapped to give one element per line with proper indentation.
Wrap text	Select this check box to have long lines wrapped according to the code style settings.
Align attributes	Select this check box to have attributes in sequential lines aligned.
Keep white spaces	When this check box is selected, the editor preserves all whitespaces within tags. The same refers also to the indents, and line breaks.
Spaces	In this area, define the usage of spaces for attributes and tag names. – Around "=" in attribute: select this check box to have spaces added around the "=" symbol in attributes. – After tag name: select this check box to have spaces added after tag names. – In empty tag: select this check box to have spaces added in empty tags.
CDATA	In this area, define the usage of whitespaces around and inside CDATA sections in MXML files: – Whitespaces around: from the drop-down list, choose how whitespaces around `CDATA` will be treated. – Preserve: all whitespaces will be left intact after reformatting. – Remove (keep with tags): all whitespaces around `CDATA` will be removed, and tags will be kept on the same lines. – New lines: new lines will be added before and after `CDATA` .

- Keep whitespaces inside: If this check box is selected, whitespaces will be preserved after `CDATA[` and before `]]`.

Arrangement

This tab lets you define a set of rules that rearranges your code according to your preferences.

Item	Description
Matching rules	Use this area to define elements order as a list of rules, where every rule has a set of matches such as modifier or type. – + - use this button to add a rule. The empty rule dialog window opens. – — - use this button to remove the rule from the list. – - use this button to edit an existing rule. To see this button, navigate to the rule that you want to edit and click on the button. In pop-up window that opens, modify the rule fields. – ↑↓ - use these buttons to move the selected rule up or down.
Empty rule	Use this window to create a new matching rule or edit an existing one. You can select from the following filters: – Type - use this filter to choose classes or methods for your rule. Note that clicking twice on the type keyword will negate the condition. – Name - use this field to specify entry names in the rule. This filter matches only entry names such as field names, method names, class names and etc. The filter supports regular expressions and uses a standard syntax. The match is performed against the entire name. – Namespace - use this field to specify the namespace in the rule. It lets you specify a rule that controls a namespace attribute position. – Order - use this drop-down list to select the sorting order for the rule. This option is useful when more than one element uses the same matching rule. In this case, selecting Keep order will keep the same order as was set before the rearrangement and selecting Order by Name will sort the elements with the same matching rule by their names. – Aliases - this option displays aliases that were defined in the Rules Alias Definition dialog. You can remove the ones you do not need.
	This icon appears when you select Order by Name in Order option. The icon indicates that the items in this rule are alphabetized.
Additional Settings	Use this area to set additional arrangement options. The Force rearrange drop-down list lets you select options that affect the Rearrange entries check box in the Reformat Code dialog. You can select from the following options: – Use current mode (toggled in the Reformat Code dialog) - In this case the Rearrange entries check box stays active and you can modify it in the Reformat Code dialog. – Always - In this case the Rearrange entries check box is selected and becomes read-only. – Never - In this case the Rearrange entries check box is cleared and becomes read-only.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Code Style. YAML

File | Settings | Editor | Code Style | YAML for Windows and Linux

PhpStorm | Preferences | Editor | Code Style | YAML for OS X

Use this page to configure formatting options for YAML files. View the result in the Preview pane on the right.

On this page:

- Tabs and Indents
- Set from...

Tabs and Indents

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the `Tab` key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Set from...

Click this link to reveal the list of languages to be used as the base for the current language code style. So doing, only the settings that are applicable to the current language are taken. All the other settings are not affected.

This link appears in the upper-right corner of the language-specific code style page, when applicable.

Click Reset to discard changes and return to the initial set of code style settings.

Other File Types

File | Settings | Code Style | Other File Types for Windows and Linux

PhpStorm | Preferences | Code Style | Other File Types for OS X

Ctrl+Alt+S

On this page:

- Scheme
- Text files and unsupported file types

Scheme

In this area, choose the code style scheme and change it as required. Settings of a code style scheme are automatically applied every time PhpStorm generates, refactors, or reformats your code.

The code styles are defined at the project level and at the global level.

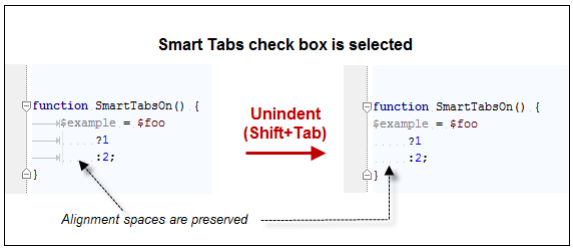
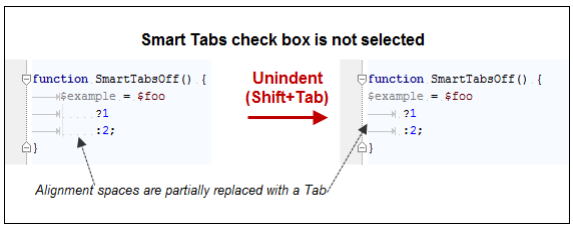
- At the Project level, settings are grouped under the Project scheme, which is predefined and is marked in bold font. The Project style scheme applies to the current project only.
  The default scheme cannot be changed! When any modifications are made, PhpStorm automatically creates a copy of the default scheme. If you want to use a different project level scheme, choose the desired scheme and copy its settings to project.
- At the Global level, settings are grouped under the predefined Default scheme, and any other scheme created by the user. The global settings are used when the user doesn't want to keep code style settings with the project and share them.

Item	Description
Scheme	From this drop-down list, select the scheme to be used. The predefined schemes are shown in bold. The custom schemes, ones created as copies of the predefined schemes, are in plain text.
Manage	Click this button to open the Code Style Schemes dialog that lets you manage the schemes.
Reset	This link appears in the upper-right corner of the dialog when the current style or scheme are modified. Click this button to discard changes.

Text files and unsupported file types

Item	Description

Use tab character	- If this check box is selected, tab characters are used:
	  - On pressing the Tab key
	  - For indentation
	  - For code reformatting
	- When the check box is cleared, PhpStorm uses spaces instead of tabs.

Smart tabs	- If this check box is selected, PhpStorm inserts tabs for indentation and reformatting. PhpStorm inserts tabs in the beginning of each line until a symbol other than the tab is inserted. After that when you press the tab key, PhpStorm will enter spaces.  - If this check box is cleared, only tabs are used. This means that a group of spaces that fits the specified tab size is automatically replaced with a tab, which may result in breaking fine alignment.  The Smart Tabs check box is available if the Use Tab Character check box is selected.
Tab size	In this text box, specify the number of spaces included in a tab.
Indent	In this text box, specify the number of spaces (or tabs if the Use Tab Character check box is selected) to be inserted for each indent level.
Indent all chained calls in a group	The check box is available only when the Indent chained methods check box is selected.

Inspections

File | Settings | Editor | Inspections for Windows and Linux

PhpStorm | Preferences | Editor | Inspections for OS X

`Ctrl+Alt+S`

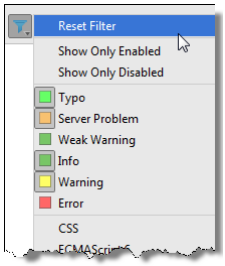
Use this page to customize inspection profiles , configure inspection severities , disable and enable inspections , and configure inspections for different scopes . The page is divided into the following areas:

- Profile management
- Toolbar
- Inspection severity and scopes
- Options

Profile management

Item	Description
Profile	From this drop-down list, select the name of the profile to configure. All modified inspections are highlighted. Note that the selected profile is automatically used for project highlighting after clicking Apply.
Manage	Click this button to reveal the following submenu:
	Set as Global/Copy to Project: For the project level profiles, choose the command Set as Global to move the selected profile to the global level. For the global level profiles, choose the command Copy to Project to create the project level duplicate of the selected profile.
	Copy: Choose this command to create a copy, based on the current profile .
	Rename: Choose this command to change the name of the current profile in the Profile field.
	Delete: Choose this command to delete the current profile. The pre-defined profiles cannot be deleted, so this command is only available for the user-defined profiles.
	Export: Choose this command to export the selected profile as an `xml` file.
	Import: Choose this command to import a profile from an `xml` file.

Toolbar

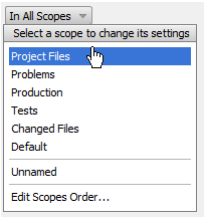
Item	Tooltip and Shortcut	Description
in		Use this text box to search through the list of inspections. In the search field, start typing the desired inspection name, or any characters contained in the inspection name or description. PhpStorm shows the list of matching occurrences. As you type a search string, the matching inspections are highlighted. To finalize the search, press `Enter` . The used search strings are memorized in the history list. – : Click this button to reveal the history list. – : Click this button to clear the search history.
	Filter Inspections	Click this button to show the list of available filters:  Click the desired filters to reduce the list. The command Reset Filters becomes available, if some of the filters are checked.

/	Expand All/Collapse All
	`Ctrl+NumPad Plus`
	`Ctrl+NumPad -`
		Click these buttons to have all inspection nodes expanded/collapsed.

	Reset to Empty	Click this button to have all the check boxes of the profile cleared and thus disable all the profile inspections.
	Advanced Settings	Click this button the show the menu with the following check commands: – Disable new inspections by default: select this check box, if you don't want the new inspections that appear after PhpStorm update, to become available. – Reset to Default settings: select this check box to discard all changes.

Inspection severity and scopes

Item	Description
Description	This read-only field shows the description of the selected inspection.
Inspection severity	From this drop-down list, select the desired severity to assign to the current inspection. Refer to the section Configuring Inspection Severities for details.
Scopes	Click this drop-down list to reveal the list of available scopes:



Clicking a scope in the list results in showing the scopes toolbar:

Alt+Insert	Click this button to specify the scope for the selected inspection.
Alt+Delete	Click this button to delete the selected scope for the current inspection.

Choosing the option Edit Scopes Order results in showing the Scopes Order dialog box, where one can change the order of scopes using the up and down arrows ( ) or keyboard shortcuts (Alt+Up / Alt+Down).

It is possible to select several inspections and add/remove scopes for the entire selection.

Options

This area is only available for some types of inspections, provided that an inspection of this type is enabled (the check box next to it is selected). Use the controls in this area to re-configure the default inspection settings.

File and Code Templates

File | Settings | Editor | File and Code Templates for Windows and Linux

PhpStorm | Preferences | Editor | File and Code Templates for OS X

`Ctrl+Alt+S`

Files can be created according to pre-defined templates (file templates). Use this page to view, edit, and create such templates.

Different groups of templates are located on different tabs .

When you select a template, its contents and description are displayed in the right-hand part of the page.

- Per-project vs default scheme
- Tabs
- Toolbar
- Template settings and contents

Per-project vs default scheme

Item	Description
Scheme	From this drop-down list, choose whether file and code template settings pertain to the entire workspace, or the current project: – Default scheme is selected, when file and code templates are global. – Project scheme is selected, if you want to use the sharable project-specific file and code templates.

Refer to the section Project and IDE Settings for details.

Tabs

Tab	Description
Files	This tab displays the available file templates. You can edit the existing templates, or create new ones. Note The templates shown in bold font cannot be deleted; their names and extensions cannot be edited.
Includes	This tab shows the templates for reusable fragments that can be included in file templates. You can edit the existing templates, or create new ones. PhpStorm provides as set of predefined built-in templates for generating PHP file headers and PHPDoc comments for PHP classes, fields, and functions (methods).
Code	This tab displays built-in snippets, i.e. templates for code fragments that PhpStorm can generate in various typical situations, for example, for generating implemented or overridden method bodies. You can edit the existing snippets, but you cannot create new ones.

Toolbar

Item	Tooltip	Description
+	Create Template	Click this button to create a new template in the currently opened tab. This option is only available in the Files and the Includes tabs. The location of the new template is defined by the Schema drop-down list. If
–	Remove Template	Click this button to delete the selected template. This option is only available for certain templates in the Files and the Includes tabs.
	Copy Template	Click this button to create a copy of the selected template. This option is only available in the Files and the Includes tabs.
	Reset to Default	Click this button to revert the selected template to its original state. This option is only available for templates that have been modified (they are highlighted in blue).
N/A	Reset	This link appears in the top-right corner of the page when you start editing a template. Clicking this link resets all unsaved changes to any template in any tab.

Template settings and contents

Item	Description
Name	This text box appears when a new template is created. Specify the name of the new template.
Extension	In this text box, specify the extension. PhpStorm will apply this template when new files of this type are created.
Template text	Edit the template contents. You can use: – Plain text. – `#parse` directives to work with template includes . – Custom variables. Their names can be defined right in the template through the `#set` directive or will be defined during the file creation. – Variables to be expanded into corresponding values in the `${<variable_name>}` format. The available predefined file template variables are: – `${PROJECT_NAME}` - the name of the current project. – `${FILE_NAME}` - the name of the PHP file that will be created. – `${NAME}` - the name of the new file which you specify in the New File dialog box during the file creation.

- `${USER}` - the login name of the current user.
- `${DATE}` - the current system date.
- `${TIME}` - the current system time.
- `${YEAR}` - the current year.
- `${MONTH}` - the current month.
- `${DAY}` - the current day of the month.
- `${HOUR}` - the current hour.
- `${MINUTE}` - the current minute.
- `${PRODUCT_NAME}` - the name of the IDE in which the file will be created.
- `${MONTH_NAME_SHORT}` - the first 3 letters of the month name. Example: Jan, Feb, etc.
- `${MONTH_NAME_FULL}` - full name of a month. Example: January, February, etc.

PhpStorm provides a set of additional variables for PHP include templates , that is, templates of reusable fragments that can be included in other PHP file template. The built-in PHP include templates are intended for generating file headers and PHPDoc documentation comments . The following variables are available in PHP include templates:

- `${NAME}` - the name of the class, field, or function (method) for which the PHPDoc comment will be generated.
- `${NAMESPACE}` - the fully qualified name (without a leading slash) of the class or field namespace.
- `${CLASS_NAME}` - the name of the class where the field to generate the PHPDoc comment for is defined.
- `${STATIC}` - gets the value `static` if the function (method) or field to generate the comment for is `static` . Otherwise evaluates to an `empty string` .
- `${TYPE_HINT}` - a prompt for the `return` value of the function (method) to generate the comment for. If the return type cannot be detected through the static analysis of the function (method), evaluates to `void` .
- `${PARAM_DOC}` - - a documentation comment for parameters. Evaluates to a set of lines `@param type name` . If the function to generate comments for does not contain any parameters, the variable evaluates to empty content.
- `${THROWS_DOC}` - a documentation comment for exceptions. Evaluates to a set of lines `@throws type` . If the function to generate comments for does not throw any exceptions, the variable evaluates to empty content.
- `${DS}` - a dollar character (`$`). The variable evaluates to a plain dollar character (`$`) and is used when you need to escape this symbol so it is not treated as a prefix of a variable.
- `${CARET}` - indicated the position of the caret after generating and adding the comment.
  This `${CARET}` variable is applied only when a PHPDoc comment is generated and inserted during file creation. When a PHPDoc comment is created through Code | Generate | PHPDoc block, multiple selection of functions or methods is available so documentation comments can be created to several classes, functions, methods, or fields. As a result, PhpStorm cannot "choose" the block to apply the `${CARET}` variable in, therefore in this case the `${CARET}` variable is ignored.

Treating dollar sign

- You can prevent treating dollar characters (`$`) in template variables as prefixes. If you need a dollar character (`$` inserted as is, use the `${DS}` file template variable instead. When the template is applied, this variable evaluates to a plain dollar character (`$`).
  Examples:
  - To use some version control keywords (such as `$Revision$` , `$Date$` , etc.) in your default class template, write `${DS}` instead of the dollar prefix (`$`).
  - The template code `${DS}this` will be rendered as `$this` .

Note PhpStorm doesn't prompt for the values of Velocity variables defined with `#set` .

Reformat according to style	Select this check box, to have PhpStorm reformat generated stub files according to the style defined on the Code Style page . This option is only available in the Files tab.
Enable Live Templates	Select this check box to use a live template inside a file template. So doing, one has to put the live template fragments into Velocity escape syntax. For example: `#[[ $MY_VARIABLE$ $END$ ]]#` Thus, one can specify the cursor position. Note that it is required to use the live template variables here!
Description	This read-only field provides information about the template, its predefined variables, and the way they work. This field is not available in custom templates.

File Encodings

File | Settings | Editor | File Encodings for Windows and Linux

PhpStorm | Preferences | Editor | File Encodings for OS X

Ctrl+Alt+S

Use this dialog to configure encoding options for a project and for the entire IDE.

The file or directory encodings take precedence over the project encoding , which, in turn, takes precedence over the IDE encoding .

If the file or directory encodings are not defined, then the project encoding is taken. If the project encoding cannot be taken (for example, if a project is not yet created), then the IDE encoding is taken.

Item	Description
IDE Encoding	From this drop-down list, choose the encoding to be used when no project is currently opened. The encoding will be applied, for example, when you specify settings of a default project or check out sources from a version control storage. Choose System Default to have the default encoding of your operating system used or choose a specific encoding.
Project Encoding	From this drop-down box, choose the default encoding to use in the folders for which no encoding is appointed in the Default encoding field below. Choose System Default to have the default encoding of your operating system used or choose a specific encoding.
File/Directory	This column displays the project tree view.
Default encoding	This column displays encoding for a file or directory, if applicable. If encoding is defined within a file, it cannot be configured, and is shown in grey font. If encoding is configurable, click the Default Encoding column for a selected file or directory, and choose encoding from the drop-down list. Encoding information embedded in a file overrides the selected one; encoding information for the nested files or directories overrides that for the outer directories or the whole project.
Default encoding for properties files	Use this drop-down list to define encoding for the properties files in the project.
Transparent native-to-ascii conversion	Select this option to show in properties files the national characters (non-ISO 8859-1), stored as escape sequences. If this check box is not selected, the national characters will not be shown.

Live Templates

File | Settings | Editor | Live Templates for Windows and Linux

PhpStorm | Preferences | Editor | Live Templates for OS X

`Ctrl+Alt+S`

Use this page to create, manage, and edit live templates .

On this page:

- List of available live templates
- Context menu of a live template
- Template editing area
- Side note about predefined template variables
- Predefined functions to be used in live template variables

List of available live templates

Item	Tooltip and shortcut	Description
By default expand with		Use this drop-down list to specify the default invocation key for all templates. Individual expansion keys for the particular templates are defined in the editing area . If the standard expansion keys (Tab, Enter, or Space) are not desirable, select the Custom option from this drop-down list. When Custom is selected, the Change link appears next to the drop-down, leading you to the Keymap page.
Live Templates		This list shows all currently available template abbreviations supplied with their descriptions. The abbreviations are grouped below nodes and sorted alphabetically within each group. To activate a template or an entire group, select the check box to its left. Note – Only active templates are displayed upon invoking live templates in the editor. – If a template is active, the editor is sensitive to its abbreviation. Otherwise, the abbreviation is considered merely a set of characters.
+	Add `Alt+Insert`	Click this button to have a new template item added to the current group of template. You can define the template abbreviation, description, text, variables, expansion key, and context in the editing area below .
−	Remove `Delete`	Click this button to have the selected live template removed from the list.
	Duplicate	Click this button to create a new template based on the selected template. A new template item is added to the current node and the fields in the Template Text area show the definition of the selected template.
	Restore	Click this button to restore the deleted live templates. This button is only enabled when the changes are applied.

Context menu of a live template

Item	Description
Move	Choose a group to move the selected template to.
Change context	Choose this command to modify the set of contexts where the current template is enabled. Upon choosing this command, a list of supported language contexts is displayed. To make PhpStorm consider a context sensitive to the template, select a check box next to the context name. The available context types depend on the enabled plugins.
Copy	Choose this command to create a serialized template XML in the system clipboard. Refer to section Sharing Live Templates .
Paste	Choose this command to paste an XML representation of the copied templates to the selected group of templates . Refer to section Sharing Live Templates .
Restore defaults	This command only appears on the context menus of the modified templates, marked blue. Choose this command to restore the default template settings.

Template editing area

The focus is moved to this area in the following cases:

- When you click the Add + or Copy button.
- When you select a live template in the list.
- When you select a fragment of code in the editor and choose Tools | Save as Live Template .

Use controls of this area to create new live templates and edit the settings for the existing ones.

You can navigate through the Template Text Area using the hot keys that are marked in the field labels.

Item	Description

Abbreviation	In this text box, specify the template abbreviation .
Description	In this text box, provide optional description of a template or an example of its usage.
Template Text	In this text box, type the template body that may contain plain text and variables in the format `$<variable name>$` . When editing the live template variables, mind the following helpful hints: – If you need a dollar sign (`$`) in the template text, escape it by duplicating this character (`$$`). – To change the variables in a template, click the Edit Variables button and configure the variables as described in Creating and Editing Template Variables . The Edit Variables button is enabled only if the template body contains at least one user-defined variable, that is, a variable different from `$END$` or `$SELECTION$` .

Side note about predefined template variables

PhpStorm supports two predefined live template variables: `$END$` and `$SELECTION$` .

You cannot edit the predefined live template variables `$END$` and `$SELECTION$` .

– `$END$` indicates the position of the cursor after the template is expanded. For example, the template `return $END$;` will be expanded into

```
return ;
```

with the cursor positioned right before the semicolon.

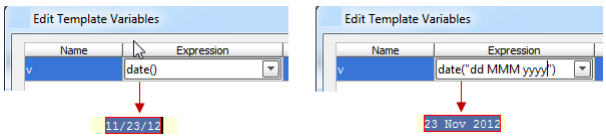
– `$SELECTION$` is used in surround templates and stands for the code fragment to be wrapped. After the template is expanded, the selected text is wrapped as specified in the template.
For example, if you select `EXAMPLE` in your code and invoke the `"$SELECTION$"` template via the assigned abbreviation or by pressing `Ctrl+Alt+T` and selecting the desired template from the list, PhpStorm will wrap the selection in double quotes as follows:

```
"EXAMPLE"
```

Applicable in:	This read-only field shows the languages and/or pieces of code where the editor should be sensitive to the template. Upon pressing `Ctrl+J` in such context, PhpStorm displays a list of templates that are valid for this context.
Change	Click this link to modify the set of contexts where the current template is enabled. Upon clicking the link, a list of supported language contexts is displayed. To make PhpStorm consider a context sensitive to the template, select a check box next to the context name. The available context types depend on the enabled plugins.
Edit Variables	Click this button to open the Edit Template Variables dialog box, where you can define how PhpStorm should process template variables upon template expansion. The Edit Variables button is enabled only if the template body contains at least one user-defined variable, that is, a variable different from `$END$` or `$SELECTION$` . The Edit Template Variables dialog box contains a complete list of available functions. See the list of predefined functions below on this page.
Options	In this area, define the behavior of the editor when a template is expanded. – Expand with - from this drop-down list, choose the key to invoke the template. – Reformat according to style - select this check box to have PhpStorm automatically reformat the expanded text according to the current style settings, defined on the Code Style page .

Predefined functions to be used in live template variables

Warning! Note that the list of predefined functions in PhpStorm depends upon the installed and enabled plugins.

Item	Description
`camelCase(String)`	Returns the string passed as a parameter, converted to camel case. For example, `my-text-file` / `my text file` / `my_text_file` will be converted to `myTextFile` .
`capitalize(String)`	Capitalizes the first letter of the name passed as a parameter.
`capitalizeAndUnderscore(sCamelCaseName)`	Capitalizes the all letters of a CamelCase name passed as a parameter, and inserts an underscore between the parts. For example, if the string passed as a parameter is `FooBar` , then the function returns `FOO_BAR` .
`classNameComplete()`	This expression substitutes for the class name completion at the variable position.
`clipboard()`	Returns the contents of the system clipboard.
`snakeCase(String)`	Returns CamelCase string out of snake_case string. For example, if the string passed as a parameter is `foo_bar` , then the function returns `fooBar` .
`complete()`	This expression substitutes for the code completion invocation at the variable position.
`completeSmart()`	This expression substitutes for the smart type completion invocation at the variable position.
`date(sDate)`	Returns the current system date in the specified format. By default, the current date is returned in the default system format. However, if you specify date format in double quotes, the date will be presented in this format: 
`decapitalize(sName)`	Replaces the first letter of the name passed as a parameter with the corresponding lowercase letter.
`enum(sCompletionString1,sCompletionString2,...)`	List of comma-delimited strings suggested for completion at the template invocation.

`escapeString(sEscapeString)`	Escapes the specified string.
`expectedType()`	Returns the type which is expected as a result of the whole template. Makes sense if the template is expanded in the right part of an assignment, after return, etc.
`fileName(sFileName)`	Returns file name with extension.
`fileNameWithoutExtension()`	Returns file name without extension.
`firstWord(sFirstWord)`	Returns the first word of the string passed as a parameter.
`lineNumber()`	Returns the current line number.
`lowercaseAndDash(String)`	Returns lower case separated by dashes, of the string passed as a parameter. For example, the string `MyExampleName` is converted to `my-example-name`.
`snakeCase(sCamelCaseText)`	Returns snake_case string out of CamelCase string passed as a parameter.
`spaceSeparated(String)`	Returns string separated with spaces out of CamelCase string passed as a parameter. For example, if the string passed as a parameter is `fooBar`, then the function returns `foo bar`.
`time(sSystemTime)`	Returns the current system time.
`timestamp()`	This function without parameters returns the current time in milliseconds from the 1st January 1970. Refer to the article Unix time .
`underscoresToCamelCase(sCamelCaseText)`	Returns the string passed as a parameter with CamelHump letters substituting for underscores. For example, if the string passed as a parameter is `foo_bar`, then the function returns `fooBar`.
`underscoresToSpaces(sParameterWithSpaces)`	Returns the string passed as a parameter with spaces substituting for underscores.
`user()`	Returns the name of the current user.
`JsArrayVariable`	Returns JavaScript array name.
`jsClassName()`	Returns the name of the current JavaScript class.
`jsComponentType`	Returns the JavaScript component type.
`jsMethodName()`	Returns the name of the current JavaScript method.
`jsQualifiedClassName`	Returns the complete name of the current JavaScript class.
`jsSuggestIndexName`	Returns a suggested name for an index.
`jsSuggestVariableName`	Returns a suggested name for a variable.

Edit Template Variables Dialog

File | Settings | Live Templates - Edit Variables for Windows and Linux

PhpStorm | Preferences | Live Templates - Edit Variables for OS X

`Ctrl+Alt+S`

The dialog opens when you click the Edit Variables button in the Template Text area on the Live Templates page.

The Edit Variables button is enabled only if the template body contains at least one user-defined variable, that is, a variable different from `$END$` or `$SELECTION$`.

Use this dialog box to create and edit expressions for variables in the selected live template.

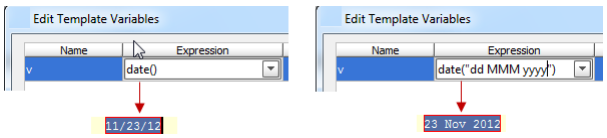
On this page:

- Controls
- Predefined Functions to Use in Live Template Variables

Controls

Item	Description
Name	In this text box, view or edit the variable name in the format `$<variable_name>$`.
Expression	In this text box, specify the expression to have the value of the corresponding template input field calculated automatically. This expression may contain constructs of the following basic types: – String constants in double quotes. – The name of another variable defined in a live template. – Predefined functions with possible arguments. Type an expression manually or select a predefined function from the drop-down list. The list shows also the number and type of parameters, if any, for the selected function. The available functions are listed alphabetically in the Functions table.
Default value	In this text box, specify the default string to be entered in the corresponding input field of the expanded template, if the expression does not give any result after calculation. Note that a default value of a variable is an expression that can refer to other live template variables. To define the default value as a literal, enclose it in quotation marks.
Skip if defined	Select this check box to have PhpStorm proceed with the next input field, if the value of the current input field is defined.
Move Up / Move Down	Use these buttons to change the order of variables in the list. The order of variables in the table determines the order in which PhpStorm will switch between the corresponding input fields when the template is expanded.

Predefined Functions to Use in Live Template Variables

Item	Description
`camelCase(String)`	Returns the string passed as a parameter, converted to camel case. For example, `my-text-file` / `my text file` / `my_text_file` will be converted to `myTextFile`.
`capitalize(String)`	Capitalizes the first letter of the name passed as a parameter.
`capitalizeAndUnderscore(sCamelCaseName)`	Capitalizes the all letters of a CamelCase name passed as a parameter, and inserts an underscore between the parts. For example, if the string passed as a parameter is `FooBar`, then the function returns `FOO_BAR`.
`classNameComplete()`	This expression substitutes for the class name completion at the variable position.
`clipboard()`	Returns the contents of the system clipboard.
`snakeCase(String)`	Returns CamelCase string out of snake_case string. For example, if the string passed as a parameter is `foo_bar`, then the function returns `fooBar`.
`complete()`	This expression substitutes for the code completion invocation at the variable position.
`completeSmart()`	This expression substitutes for the smart type completion invocation at the variable position.
`date(sDate)`	Returns the current system date in the specified format. By default, the current date is returned in the default system format. However, if you specify date format in double quotes, the date will be presented in this format: 
`decapitalize(sName)`	Replaces the first letter of the name passed as a parameter with the corresponding lowercase letter.
`enum(sCompletionString1,sCompletionString2,...)`	List of comma-delimited strings suggested for completion at the template invocation.
`escapeString(sEscapeString)`	Escapes the specified string.
`expectedType()`	Returns the type which is expected as a result of the whole template. Makes sense if the template is expanded in the right part of an assignment, after return, etc.
`fileName(sFileName)`	Returns file name with extension.
`fileNameWithoutExtension()`	Returns file name without extension.
`firstWord(sFirstWord)`	Returns the first word of the string passed as a parameter.

`lineNumber()`	Returns the current line number.
`lowercaseAndDash(String)`	Returns lower case separated by dashes, of the string passed as a parameter. For example, the string `MyExampleName` is converted to `my-example-name`.
`snakeCase(sCamelCaseText)`	Returns snake_case string out of CamelCase string passed as a parameter.
`spaceSeparated(String)`	Returns string separated with spaces out of CamelCase string passed as a parameter. For example, if the string passed as a parameter is `fooBar`, then the function returns `foo bar`.
`time(sSystemTime)`	Returns the current system time.
`timestamp()`	This function without parameters returns the current time in milliseconds from the 1st January 1970. Refer to the article Unix time.
`underscoresToCamelCase(sCamelCaseText)`	Returns the string passed as a parameter with CamelHump letters substituting for underscores. For example, if the string passed as a parameter is `foo_bar`, then the function returns `fooBar`.
`underscoresToSpaces(sParameterWithSpaces)`	Returns the string passed as a parameter with spaces substituting for underscores.
`user()`	Returns the name of the current user.
`JsArrayVariable`	Returns JavaScript array name.
`jsClassName()`	Returns the name of the current JavaScript class.
`jsComponentType`	Returns the JavaScript component type.
`jsMethodName()`	Returns the name of the current JavaScript method.
`jsQualifiedClassName`	Returns the complete name of the current JavaScript class.
`jsSuggestIndexName`	Returns a suggested name for an index.
`jsSuggestVariableName`	Returns a suggested name for a variable.

File Types

File | Settings | Editor | File Types for Windows and Linux

PhpStorm | Preferences | Editor | File Types for OS X

Ctrl+Alt+S

Use this page to manage the list of file types and extension patterns to be recognized by PhpStorm.

Item	Keyboard Shortcut	Description
Recognized file types		This list box displays all the default and custom file types currently supported by PhpStorm. Refer to the section File Types Recognized by PhpStorm . Use the Add, Edit, and Remove buttons to manage the contents of the list box. Tip Default types cannot be edited or removed.
+	Alt+Insert	Click this button to open the New File Type dialog and define a new custom file type there.
	Enter	Click this button to open the Edit File Type dialog box and edit the selected file type there. This button is disabled when a default file type is selected.
−	Alt+Delete	Click this button to delete the selected file type from the list. This button is disabled when a default file type is selected.
Registered Patterns		Shown in this area are all the registered extensions associated with the file type selected in the Recognized file types list. Use +, and − to manage the corresponding patterns.
+	Alt+Insert	Use this icon or shortcut to open the Add wildcard dialog box and specify a new pattern using wildcards there.
	Enter	Use this icon or shortcut to edit the selected pattern.
−	Alt+Delete	Use this icon or shortcut to remove the selected pattern from the list.
Ignore files and folders		In this text box, specify the files and folders, which you want to be ignored by PhpStorm. Such files and folders will be completely excluded from any kind of processing. By default the list includes temporary files, service files related to version control systems, etc. You can specify multiple names or wildcard masks, with semicolons (`;`) as separators. Below is the default setting, in case you need to restore it: `CVS;SCCS;RCS;rcs;.DS_Store;.svn;.pyc;.pyo;*.pyc;*.pyo;.git;*.hprof;_svn;.hg;*.lib;*~;__pycache__;.bundle;*.rbc;*$py.class;`

New File Type

File | Settings | Editor | File Types - Add/Edit for Windows and Linux

PhpStorm | Preferences | Editor | File Types - Add/Edit for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Add button or select a custom file type and click the Edit button on the File Types page .

Use the dialog box to configure and re-configure presentation and highlighting of keywords, comments, numbers etc. in files of a specific custom file type. These settings make the basis for parsing files of this type in the editor.

Item	Description
Name	In this text box, specify the name of the file type.
Description	In this text box, provide an optional description of the file type.
Syntax Highlighting	In this area, specify the character strings to indicate borders of comments, the numeric system used, and configure highlighting for brackets, braces, etc. syntax elements.
Line comment	In this text box, specify the character string to indicate the start of a single-line comment.
Only at line start	If this check box is selected, the character string that denotes the beginning of a line comment, is recognized as a comment if located in the first position of a line. This check box becomes available if at least one character is inserted in the Line comment field.
Block comment start	In this text box, specify the character string to indicate the start of a block comment.
Block comment end	In this text box, specify the character string to indicate the end of a block comment.
Hex prefix	In this text box, specify the character string to indicate that the subsequent value is a hexadecimal number. For example, `0x` .
Number postfixes	In this text box, specify the character string to indicate which numeric system or unit is used. A postfix is a trailing string of characters, for example, `e-3, kg` etc.
Support paired braces	Select this check box, to have paired braces highlighted.
Support paired brackets	Select this check box, to have paired brackets highlighted.
Support paired parens	Select this check box, to have paired parentheses highlighted.
Support string escapes	Select this check box, to have string escapes highlighted.
Ignore case	Select this check box to have PhpStorm ignore case when processing file type extensions.
Keywords	Use this area to flexibly configure highlighting of keywords by grouping them into sets and associating each set with its own highlighting scheme . The area consists of 4 tabs. In each tab, create a set of keywords using the Add and Remove buttons. To associate a keyword set 1-4 with a highlighting scheme, edit the corresponding Keyword1 - Keyword4 property on the Custom page of the Colors and Fonts dialog box.
+	Click this button to open the Add a new keyword dialog box and define the new keyword there.
−	Click this button to delete the selected keyword from the list.

Copyright

File | Settings | Editor | Copyright for Windows and Linux

PhpStorm | Preferences | Editor | Copyright for OS X

`Ctrl+Alt+S`

Use this page to associate copyright profiles with the specific scopes within your project.

In this section:

- Copyright
  - Copyright options
- Copyright Profiles
- Formatting
  - File Types

Copyright options

Item	Tooltip and shortcut	Description
Default project copyright		Use this drop-down list to specify the default copyright profile at the project level. The default profile applies to those project files that do not belong to any defined scope. The default profile also applies when no specific profile-scope combinations have been created. By default, PhpStorm suggests No copyright. If No copyright value is selected, then, on an attempt to create a copyright notice in a file out of any defined scope, PhpStorm will show a dialog box asking you whether you want to edit the copyright settings. Choosing OK results in opening this page, where you can select the default project profile from the existing ones, or create a new one; choosing Cancel generates no copyright notice at all. The drop-down list contains the available copyright profiles defined in the Copyright Profiles page. If only No copyright option is available, the scope-profile combinations area is disabled. With at least one copyright profile defined, the scope-profile combinations area below becomes enabled.
Scope-profile combinations area		Note that unless you have at least one copyright profile and scope , the controls below are disabled.
Scope		Click an entry in this column to select the desired shared scope from the drop-down list. If the desired scope is not yet defined, click the link below to open the Scopes dialog box, where you can create a new scope or edit an existing one.
Copyright		Click an entry in this column to select a copyright profile from the drop-down list of available copyright profiles.
+	Add `Alt+Insert`	Use this icon or shortcut to add a new scope-profile combination.
−	Delete `Alt+Delete`	Use this icon or shortcut to remove the selected item from the list.
↑↓	Up/Down `Alt+Up` / `Alt+Down`	Use these icons or shortcuts to move the selected item one line up or down in the list. The order of scope-profile combination is important, because if a file belongs to several scopes, these scopes are checked upside down. So doing, the first scope where this file belongs, is declared the proper one, and its copyright profile is used.
Select Scopes to add new scopes or modify existing ones		Click this link to open the Scopes page , where you can create a new scope or edit an existing one.

Copyright Profiles

File | Settings | Editor | Copyright | Copyright Profiles for Windows and Linux

PhpStorm | Preferences | Editor | Copyright | Copyright Profiles for OS X

`Ctrl+Alt+S`

Use this page to create, edit, and remove copyright profiles.

In this topic:

– Profiles Toolbar

– Copyright Profile Page

Profiles Toolbar

Item	Tooltip and Shortcut	Description
+	`Insert`	Click this button to add a new profile at the desired level. You can choose to create an empty profile or a profile based on the current default profile.
–	`Delete`	Click this button to discard the selected profile.
	`Ctrl+D`	Click this button to open the Copy Copyright Profile dialog box, where you can create a copy of the selected profile.
	Import	Click this button to import a file that contains the desired copyright notice definition.

Tip You can perform the same actions using context menus of the profile nodes.

Copyright Profile Page

Use this page to configure the selected profile: define the copyright notice to be generated and specify the keyword to detect copyright notices in comments.

Item	Description
Name	Use this text field to view or edit the name of the selected copyright profile.
Copyright text (may contain Velocity templates)	Use this text area to view or edit the copyright notice to be generated. A copyright profile can contain an explicit plain text of the copyright notice or its definition through a Velocity template. Currently the following variables are available in the Velocity context:

Name	Type	Comment
`$today`	`DateInfo`	The current date and time.
`$file.fileName`	`String`	The name of the currently opened file where the notice is to be generated.
`$file.pathName`	`String`	The complete path and name of the currently opened file where the notice is to be generated.
`$file.className`	`String`	The name of the currently opened Java file where the notice is to be generated.
`$file.qualifiedClassName`	`String`	The fully qualified name of the currently opened Java file where the notice is to be generated.
`$file.lastModified`	`DataInfo`	The date and time when the current file was last changed.
`$project.name`	`String`	The name of the current project.
`$module.name`	`String`	The name of the current module.
`$username`	`String`	The name of the current user.
`DateInfo` has the following properties:		
`year`	`int`	The current year.
`month`	`int`	The current month (1-12).
`day`	`int`	The current date of month (1-31).
`hour`	`int`	The current hour (0-11).
`hour24`	`int`	The current hour (0-23).
`minute`	`int`	The current minute of the hour (0-59).
`second`	`int`	The current second of the minute (0-59).
`DateInfo` has the following method:		
`format(String format)`	`String`	See java.text.SimpleDateFormat format options.

Validate	Click this button to check that the Velocity template has been specified correctly.
Regex to detect copyright in comments	Use this text box to type, view, or edit the regular expression that will be used to find copyright notices in comments. Tip This regular expression should match the above specified copyright notice. Otherwise instead of updating copyright notices, PhpStorm will insert new ones.

Allow replacing copyright if old copyright contains:	Use this text box to type or edit the string that will be recognized in the existing copyright notice, and allow replacing it with the new one.

Formatting

File | Settings | Editor | Copyright | Formatting for Windows and Linux

PhpStorm | Preferences | Editor | Copyright | Formatting for OS X

Ctrl+Alt+S

Use this page to configure common formatting options, regardless of the type of a particular file.

Item	Description
Comment Type	In this area, specify the type of comment to enclose copyright notices in. The available options are: – Use block comment - select this option to have copyright notices enclosed in block comments. – Prefix each line - select this check box to have each line of a copyright notice prepended with the character defined in the Separator char text box. By default, an asterisk is used. – Use line comment - select this option to have copyright notices represented as a sequence of line comments.
Borders	In this area, define how to separate copyright notices from other comments. – Separator before - select this check box have a copyright notice prepended with a line of characters defined in the Separator char text box. The number of characters in a separator line is defined in the Length text box. – Separator after - select this check box have a copyright notice followed by a line of characters defined in the Separator char text box. The number of characters in a separator line is defined in the Length text box. – Separator char - in this text box, type the character that will be used in the separator strings and as a prefix or ending character in block comments. – Box - select this check box to have each line of a copyright notice followed by a character defined in the Separator char text box. – Add blank line after - select this check box to have a blank line inserted after a copyright notice.
Relative Location	In this area, specify the location of a copyright notice relative to other comments in a file. – Before other comments - when this option is selected, copyright notices are inserted above other comments. – After other comments - when this option is selected, copyright notices are inserted below other comments.
Preview	Use this area to view a sample copyright notice with the formatting you have defined.

File Types

File | Settings | Editor | Copyright | Formatting | <File Type> for Windows and Linux

PhpStorm | Preferences | Editor | Copyright | Formatting | <File Type> for OS X

Ctrl+Alt+S

The page opens when you click a file type below the Formatting node. Use this page to define formatting of copyright notices depending on the target file type.

Item	Description
No copyright	If this option is selected, the copyright notice will not be updated in the files of the selected type.
Use default settings	If this option is selected, the copyright notice will be updated according to the default project settings.
Use custom formatting options	If this option is selected, the copyright notice in the files of the selected type will be updated according to the custom settings defined below.
Comment Type	In this area, specify the type of comment to enclose copyright notices in. The available options are: – Use block comment - select this option to have copyright notices enclosed in block comments. – Prefix each line - select this check box to have each line of a copyright notice prepended with the character defined in the Separator char text box. By default, an asterisk is used. – Use line comment - select this option to have copyright notices represented as a sequence of line comments.
Borders	In this area, define how to separate copyright notices from other comments. – Separator before - select this check box have a copyright notice prepended with a line of characters defined in the Separator char text box. – Separator after - select this check box have a copyright notice followed by a line of characters defined in the Separator char text box. – Separator char - in this text box, type the character that will be used in the separator strings and as a prefix or ending character in block comments. – Length - in this text box, type the number of characters in a spartor line. – Box - select this check box to have each line of a copyright notice followed by a character defined in the Separator char text box. – Add blank line after - select this check box to have a blank line inserted after a copyright notice.
Relative Location	In this area, specify the location of a copyright notice relative to other comments. – Before other comments - when this option is selected, copyright notices are inserted above other comments. – After other comments - when this option is selected, copyright notices are inserted below other comments.
Preview	Use this area to view a sample copyright notice with the formatting you have defined.

Location in File	Use this area to specify the location of the copyright notice in a file. Depending on the file type, the available options are:
	– HTML, XML
	  – Before the root tag
	  – Before the Doctype statement
	– For PHP, JavaScript, CSS, SCSS, and SASS file types no choice is available. Copyright notices are always inserted at the top of a file.

Emmet

File | Settings | Editor | Emmet for Windows and Linux

PhpStorm | Preferences | Editor | Emmet for OS X

`Ctrl+Alt+S`

On the nested pages, enable PhpStorm to use Emmet in HTML, XML , JavaScript (JSX Harmony) files and style sheets .

Item	Description
Expand abbreviation with	Use this drop-down box to select the default key to expand Emmet selectors with. This key will also by default expand Emmet live templates . **Note** PhpStorm expands abbreviations only if their output does not exceed 15 KB.

CSS

File | Settings | Editor | Emmet - CSS for Windows and Linux

for OS X

Ctrl+Alt+S

Item	Description
Enable CSS Emmet	Select this check box to enable Emmet support for style sheets. If this check box is not selected, the complicated abbreviations, like `bdci:n` expanding into `border-corner-image: none;` , will not work in the editor.
Enable fuzzy search among CSS abbreviations	When this check box is selected, every unknown abbreviation will be scored against available template names. The match with the best score will be used to resolve the template. For example, with this option enabled, the following abbreviations can be equal to: – `ov:h` – `ov-h` – `o-h` – `oh`
Enable expansion of unknown properties ('unknown' to 'unknown:;')	– When this check box is selected, any entered word will be expanded into the same word followed with a colon and a semicolon; – When this check box is cleared, only known properties (for example, `color`) will be expanded this way (`color:;`)
Auto insert CSS vendor prefixes	If this check box is selected, the CSS properties listed in the table below are expanded into constructs that contain pre-pending vendor prefixes. Learn more at Vendor prefixes . If this check box is cleared, the entire table of properties is disabled.
Properties and vendor prefixes	The table contains a list of CSS properties and vendor prefixes that correspond to various browsers. – To enable or disable a property in a browser, select or clear the check box under the browser column. – To add a new property to the list, click the Add button + or press Alt+Insert . Then type the name of the property in the dialog box that opens and enable it in the relevant browsers. – To delete one or more properties from the lsit, select them and press Remove — or press Alt+Delete .

JSX

File | Settings | Editor | Emmet - JSX for Windows and Linux

PhpStorm | Preferences | Editor | Emmet - JSX for OS X

Ctrl+Alt+S

Item	Description
Enable JSX Emmet	Select this check box to use Emmet within XML fragments in the JSX Harmony context if you have chosen the JSX language level on the JavaScript page of the Settings dialog box.

HTML

File | Settings | Editor | Emmet - HTML for Windows and Linux

PhpStorm | Preferences | Editor | Emmet - HTML for OS X

Ctrl+Alt+S

ItemDescription	
Enable XML/HTML Emmet	Select this check box to enable Emmet support for XML and HTML. If this check box is not selected, complicated abbreviations, such as `div.class>ul#list>.item$)` and similar, will not work in the editor.

Enable abbreviation preview	Select this check box to have PhpStorm show a pop-up window with a preview of the entered abbreviation before actually expanding it .
	`table#users>tr.user>td`
	`<table id="users"><tr class="user"><td></td></tr></table>`
	Ctrl+Down and Ctrl+Up will move caret down and up in the editor >>

Enable automatic URL recognition while wrapping text with <a> tag	– If this check box is cleared and you attempt to wrap an URL address with the <a> tag, PhpStorm simply encloses the URL address in `<a href=""></a>` and positions the cursor inside the double quotes in the `href` attribute. For example, wrapping `http://www.jetbrains.com` will result in `<a href=" | ">http://www.jetbrains.com</a>` :
	`<a href=" | ">http://www.jetbrains.com</a>`
	– If this check box is selected and you attempt to wrap an URL address with the <a> tag, PhpStorm inserts the URL address inside the double quotes as the value of the `href` attribute and encloses the URL in `<a href="<wrapped URL>"></a>` . For example, wrapping `http://www.jetbrains.com` will result in `<a href="http://www.jetbrains.com">http://www.jetbrains.com</a>` . Moreover, PhpStorm highlights the wrapped URL green as a recognized URL:
	`<a href="http://www.jetbrains.com">http://www.jetbrains.com</a>`

Add edit point at the end of template	If this check box is selected, an editing position adds to the end of an HTML template (END); if this check box is not selected, then the new edit point is not added. Compare the following: `<tr> <td></td> <td></td> </tr>` ⟷ `<tr> <td></td> <td> </td> </tr>`
BEM	In this area, specify the BEM separators for the class names, modifiers and short elements. Refer to the Emmet documentation for details.
Filters enabled by default	In this area, specify which Emmet filters you want to be applied to an expanded abbreviation before it is shown in the editor. Learn more about filters at http://docs.emmet.io/filters/ . To have a filter always applied by default, select the check box next to it. The available options are: – XSL tuning – Comment tags – Escape – Single line – BEM – Trim line markers

Images

File | Settings | Editor | Images for Windows and Linux

PhpStorm | Preferences | Editor | Images for OS X

Ctrl+Alt+S

Specify how you want images to be shown in PhpStorm. Optionally, specify an external editor for working with images.

Item	Description
Editor	Use this area to specify the settings according to which images should be displayed in PhpStorm.
Show Grid lines by default	Select this check box to have a grid displayed when reviewing image files. Tip When the check box is selected, the area below is enabled where you can define how to display a grid and its elements.
Show Grid lines only when zoom factor equal or more than	Use this spin box to specify the minimum zoom factor to have a grid displayed.
Show Grid line after every (pixels)	Use this spin box to specify the number of pixels between a pair of grid lines.
Grid line color	From this palette, click the colour to display grid lines.
Show transparency chessboard by default	Select this check box to have transparent pieces of images shown as a chessboard.
Chessboard cell size	Use this spin box to specify the chessboard cell size.
Color of 'white' cell	From this palette, click the colour to display chessboard cells located at initially 'white' positions.
Color of 'black' cell	From this palette, click the colour to display chessboard cells located at initially 'black' positions.
Zoom image with mouse wheel (Ctrl+Mouse Wheel)	Select this check box to enable zooming the image through the Ctrl+Mouse wheel combination.
Enable smart zooming for small images	Select this check box to have small images opened with the zooming factor to the size specified below.
Preferred minimum width/height for smart zooming (pixels)	Use this spin box to set the minimum number of pixels to zoom the small images to.
External Editor	In this area, specify an external editor for working with images.
Executable path	In this text box, specify the path to the executable image editor file.

Intentions

File | Settings | Editor | Intentions for Windows and Linux

PhpStorm | Preferences | Editor | Intentions for OS X

`Ctrl+Alt+S`

Use this page to enable and disable intention actions .

In this topic:

- Intention List
- Toolbar and Controls

Intention List

The list shows all intention actions currently available in PhpStorm.

The intention actions are grouped by languages. To enable an intention action, select the check box to its left.

Toolbar and Controls

Item	Tooltip and shortcut	Description
	`Ctrl+NumPad Plus`	Expand all nodes in the intention list.
	`Ctrl+NumPad -`	Collapse all nodes in the intention list.
		Use this text box to search through the list of intention actions. As you type a search string, the intention actions that match the search pattern are displayed. To finalize the search, press `Enter`. The previously used search patterns are stored in the search history list.
		Click this button to clear the search history list.
Description		This read-only field shows the description of the selected intention action.
Usage examples		This area illustrates the effect of applying the selected intention action through the following fields: – Before - this read-only field shows an example of source code before applying the selected intention action. – After - this read-only field shows the result of applying the selected intention action to the above example of source code.

Language Injections

File | Settings | Editor | Language Injections for Windows and Linux

PhpStorm | Preferences | Editor | Language Injections for OS X

`Ctrl+Alt+S`

For the Language Injections page to be available, the IntelliLang plugin must be enabled.

Use this page to manage the list of available language injections and to configure the language injection feature for text, attributes, and parameters.

To sort the information, use the cells in the header row (Name, Language and Scope). To perform other tasks, use the check boxes and the toolbar icons.

In this section:

- Language Injections
  - Prerequisites
  - Injection entries
  - Toolbar
- Language Injection Settings Dialog: XML Tag Injection
- Language Injection Settings Dialog: XML Attribute Injection
- Language Injection Settings Dialog: Generic JavaScript
- Language Injection Settings: Generic PHP

Prerequisites

Before you start working with language injections, make sure that the IntelliLang plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Injection entries

You can sort the information by any of the columns by clicking the cells in the header row. The current sorting status is shown by the corresponding sorting marker: ▲ for the ascending order or ▼ for the descending order.

Item	Description
Leftmost column	Use the check boxes to enable or disable the corresponding injections. You can also enable or disable a number of injections at once. To do that, select the required injections in the list and click Enable Selected Injections or Disable Selected Injections on the toolbar.
Name	This column shows the names of the injections and the libraries that implement them.
Language	This column shows the languages to which the corresponding injections pertain.
Scope	One of the following: – Built-in. This is a category for pre-defined injections. In terms of the scope, those are IDE-level injections. – IDE. User-defined injections that are available in all of your projects. – Project. User-defined injections that are available only in the current project. You can move the user-defined injections between the IDE and the project levels by using on the toolbar.

Toolbar

Item	Tooltip and Shortcut	Description
+	Add `Insert`	Use this icon or shortcut to create a new entry. Select the target category from the list and then specify the settings in the dialog that opens.
–	Remove `Delete`	Use this icon or shortcut to remove the selected entries from the list.
	Edit `Enter`	Click this icon to re-configure the selected entry in the Language Injection Settings dialog box that opens.
	Duplicate	Use this icon to create a copy of the selected injection entry. Edit the copy as necessary.
	Enable Selected Injections	Click this icon to enable the injections that are currently selected in the list.
	Disable Selected Injections	Click this icon to disable the injections that are currently selected in the list.
	Move to Project/IDE Scope `Shift+Space`	Use this icon or shortcut to move the selected injections between the IDE and the project levels. See also, Scope.
	Import	Click this button to have an injection configuration from another PhpStorm installation imported. The Select Path dialog opens, where you can select the `IntelliLang.xml` file from the desired location. The dialog box that opens

		displays the entries contained in the configuration file. Remove the unnecessary entries from the list using the Delete button. This will not affect the contents of the configuration file. This selective import-feature makes it easy to share certain configurations in a team without losing any local entries as it happens when settings are imported via the core Importing Settings feature. To prevent inconsistent data, import is only possible if the existing configuration has not been changed or has been saved using the Apply button.
	Export	Click this button to have one or several selected injection configurations saved in a file. Upon clicking this button, the Export Selected Injections to File dialog box opens. − To have the configurations appended to an existing file, select the required file. − To have the configurations saved in a new file, specify the file name and choose the file type from the drop-down list.

Language Injection Settings dialog: XML Tag Injection

File | Settings | Language Injection - XML Tag for Windows and Linux

PhpStorm | Preferences | Language Injection - XML Tag for OS X

`Ctrl+Alt+S`

The dialog box opens when you click + on the Language Injections page, and choose XML Tag on the context menu, or select an entry and click .

Item	Description
Name	In this text box, type the name that will identify the injection configuration in the list. The text box is available when configuration is created (+), or configured ( ).
Language	In this area, specify the language of the injection. If necessary, define the context to be automatically made up when the injection is opened in the editor . – ID - from this drop-down list, select the ID of the language of the injection. – Prefix - in this text box, specify the character string to be automatically added before the injection. – Suffix - in this text box, specify the character string to be automatically added after the injection. The Prefix and Suffix text boxes are optional.
XML Tag	In this area, define the XML tag which indicates that the text enclosed in this tag should be treated as the selected language. – Local name - in this text box, specify the tag name without a namespace prefix. Use regular expressions to specify multiple tag names (`name1|name2`), case-insensitive names (`(?i)tagname` matches *tagname* as well as *TagName*), etc. Warning! Be sure not to enter any space characters as they would be significant for the match. – Namespace - in this text box, specify the namespace URI of the XML tag. The field is optional.
Sub-tags	Select this check box to include all the subtags recursively.
Advanced	In this area, specify additional settings to enable more fine-grained control over the injection process. – Value pattern - in this text box, type a regular expression that determines the part of the XML text's value to inject the language into. By using the first capturing group of the pattern as the target for injection, you can configure the procedure to have the language injected only into values that match a certain pattern or into multiple parts that match the pattern. Examples: `[$#]\{(.*?)\}` matches the pattern used by the JSP/JSF Expression Language. `^javascript:(.*)` matches the `javascript` protocol that can be used in hyperlink-hrefs to execute JavaScript code. – XPath condition - in this text box, specify an XPath expression to address the injection-target more precisely. The context in which the expression is evaluated is the surrounding XML tag. It is possible to use the XPath extension functions that are provided by the Jaxen XPath engine, e.g. `lower-case()` . Also, there are three additional functions that can be used to determine the current file's name, extension, and file type: `file-name()` , `file-ext()` and `file-type()` . Alternatively, a list of available functions can be retrieved through standard code completion. For the field to be active, the XPathView + XSLT Support plugin must be enabled. Warning! For performance reasons, it is recommended that you keep these expressions as simple as possible. Especially expressions that cause the whole document to be scanned, such as `//foo/bar` might cause performance problems with large files. – Single file - select this check box to specify an exact file to apply the injection to.

Language Injection Settings dialog: XML Attribute Injection

File | Settings | Language Injection - XML Attribute for Windows and Linux

PhpStorm | Preferences | Language Injection - XML Attribute for OS X

`Ctrl+Alt+S`

The dialog box opens when you click + on the Language Injections page, and choose XML Attribute Injection on the context menu, or select an entry and click .

Item	Description
Name	In this text box, type the name that will identify the injection configuration in the list. The text box is available when configuration is created (+), or configured ( ).
Language	In this area, specify the language of the injection. If necessary, define the context to be automatically made up when the injection is opened in the editor . – ID - from this drop-down list, select the ID of the language of the injection. – Prefix - in this text box, specify the character string to be automatically added before the injection. – Suffix - in this text box, specify the character string to be automatically added after the injection. The Prefix and Suffix text boxes are optional.
XML Tag	In this area, specify the XML tags in which the attributes are impacted by the defined configuration. – Local name - in this text box, specify the tag name without a namespace prefix. Use regular expressions to specify multiple tag names (`name1|name2`), case-insensitive names (`(?i)tagname` matches *tagname* as well as *TagName*), etc. Warning! Space characters are not allowed as they affect the match result. – Namespace - in this text box, specify the namespace URI of the XML tag. Both fields are optional. However, if the Local name text box is empty the configuration will apply to any attribute that matches the configured name, regardless of its containing XML tag.
XML Attribute	In this area, define the XML tag attribute which indicates that the text enclosed in a tag with such attribute should be treated as the selected language. – Local name - in this text box, specify the attribute name without a namespace prefix. Use regular expressions: For example, to match HTML event handler attributes, type `on.*` in the text box. The field is optional, unless the Local name text box in the XML Tag area is empty. If the attribute local name is not specified, the configuration applies to all attributes of the enclosing tag. – Namespace - in this text box, specify the namespace URI of the attribute.
Advanced	In this area, specify additional settings to enable more fine-grained control over the injection process. – Value pattern - in this text box, type a regular expression that determines the part of the attribute's value to inject the language into. By using the first capturing group of the pattern as the target for injection, you can configure the procedure to have the language injected only into values that match a certain pattern or into multiple parts that match the pattern. – XPath condition - in this text box, specify an XPath expression to address the injection-target more precisely. The context in which the expression is evaluated is the attribute itself. For the field to be active, the XPathView + XSLT Support plugin must be enabled. – Single file - select this check box to specify an exact file to apply the injection to.

Language Injection Settings dialog: Generic JavaScript

File | Settings | Editor | Language Injection - Generic JavaScript for Windows and Linux

PhpStorm | Preferences | Editor | Language Injection - Generic JavaScript for OS X

`Ctrl+Alt+S`

The dialog box opens when you click + and choose Generic JS on the context menu, or select an entry and click .

PhpStorm comes with a set of predefined injection configurations which is quite sufficient to ensure high productivity and comfortable environment. Therefore it is strongly recommended that you use the predefined injection configurations and avoid creating new ones.

Item	Description
Name	In this text box, type the name that will identify the injection configuration in the list. The text box is available when configuration is created (+), or configured ( ).
Language	In this area, specify the language of the injection. If necessary, define the context to be automatically made up when the injection is opened in the editor . – ID - from this drop-down list, select the ID of the language of the injection. – Prefix - in this text box, specify the character string to be automatically added before the injection. – Suffix - in this text box, specify the character string to be automatically added after the injection. The Prefix and Suffix text boxes are optional.
Places Pattern	In this text box, type the rules that define the context where you want PhpStorm recognize literals as injections. **Note** The rules are built from Program Structure Interface Patterns and are actually chained calls of methods of an internal PhpStorm language. The Program Structure Interface shows the structure of a file as PhpStorm treats it. For more information on the syntax used, refer to Custom Plugin Development guide. **Warning!** These rules are PhpStorm internals, and it is strongly recommended that you use the predefined injection configurations and avoid creating new ones.
Advanced	In this area, specify additional settings to narrow the context where the injection is applicable and thus the enable more fine-grained control over the injection process. – Value pattern - in this text box, type a regular expression that determines the context to inject the language into. By using the first capturing group of the pattern as the target for injection, you can configure the procedure to have the language injected only into values that match a certain pattern or into multiple parts that match the pattern. For example, `^javascript:(.*)` matches the `javascript` protocol that can be used in hyperlink-hrefs to execute JavaScript code. – Single file - select this check box to specify an exact file to apply the injection to. Type the name of the file in the Value pattern.

Language Injection Settings: Generic PHP

File | Settings | Language Injection - Generic PHP for Windows and Linux

PhpStorm | Preferences | Language Injection - Generic PHP for OS X

`Ctrl+Alt+S`

The dialog box opens when you click + in the Language Injection page, and choose Generic PHP on the context menu, or select an entry and click .

PhpStorm comes with a set of predefined injection configurations which is quite sufficient to ensure high productivity and comfortable environment. Therefore it is strongly recommended that you use the predefined injection configurations and avoid creating new ones.

Item	Description
Name	In this text box, type the name that will identify the injection configuration in the list. The text box is available when configuration is created (+), or configured ( ).
Language	In this area, specify the language of the injection. If necessary, define the context to be automatically made up when the injection is opened in the editor . – ID - from this drop-down list, select the ID of the language of the injection. – Prefix - in this text box, specify the character string to be automatically added before the injection. – Suffix - in this text box, specify the character string to be automatically added after the injection. The Prefix and Suffix text boxes are optional.
Places Pattern	In this text box, type the rules that define the context where you want PhpStorm recognize literals as injections. **Note** The rules are built from Program Structure Interface Patterns and are actually chained calls of methods of an internal PhpStorm language. The Program Structure Interface shows the structure of a file as PhpStorm treats it. For more information on the syntax used, refer to Custom Plugin Development guide. **Warning!** These rules are PhpStorm internals, and it is strongly recommended that you use the predefined injection configurations and avoid creating new ones.
Advanced	In this area, specify additional settings to narrow the context where the injection is applicable and thus the enable more fine-grained control over the injection process. – Value pattern - in this text box, type a regular expression that determines the context to inject the language into. By using the first capturing group of the pattern as the target for injection, you can configure the procedure to have the language injected only into values that match a certain pattern or into multiple parts that match the pattern. For example, `^javascript:(.*)` matches the `javascript` protocol that can be used in hyperlink-hrefs to execute JavaScript code. – Single file - select this check box to specify an exact file to apply the injection to. Type the name of the file in the Value pattern.

Spelling

File | Settings | Editor | Spelling for Windows and Linux

PhpStorm | Preferences | Editor | Spelling for OS X

`Ctrl+Alt+S`

Use this settings page to create your own spelling dictionaries and thus expand the basic spelling support provided by PhpStorm by default.

On this page:

- Accepted Words Tab
- Dictionaries Tab

Accepted Words Tab

Use this tab to configure the list of words that should be skipped by the Typo inspection.

Item	Tooltip and Shortcut	Description
+	`Alt+Insert`	Click this icon to open the Add New Word dialog box and specify a new entry there. CamelCase or snake_case are not supported. If you try to add a word that is already included in one of the spelling dictionaries, PhpStorm displays an error message The word <just typed word> is already in the dictionary.
−	`Alt+Delete`	Click this button to delete the selected item from the list.

Dictionaries Tab

Use this tab to configure the dictionaries to be used for spellchecking.

Item	Description
Dictionaries	This area in the bottom of the page shows a list of the dictionaries that can be used in spellchecking. The list contains the dictionaries that come bundled with PhpStorm by default and user-defined dictionaries detected in the folders from the Custom Dictionaries Folder area above. − To have a default dictionary applied in the current project, select the check box next to it. − To exclude a default dictionary from spellchecking within the scope of the current project, clear the check box next to it.
Custom Dictionaries Folder	This area displays a list of directories that contain user-defined dictionary files (text files with the `dic` extension, containing words separated with a newline). − To add a new folder to the list, click + and choose the required folder in the Select Path Dialog dialog that opens . The full path to the folder is added to the Custom Dictionaries Folder list, and all the `*.dic` files found in this folder are added to the Dictionaries list. − To remove a folder from the list, select it and click −.

TextMate Bundles

File | Settings | Editor | TextMate Bundles for Windows and Linux

PhpStorm | Preferences | Editor | TextMate Bundles for OS X

`Ctrl+Alt+S`

Use this page to import the TextMate/SublimeText 2 bundles , and to map color scheme of PhpStorm to that of TextMate.

Prerequisites

This page appears in the Settings/Preferences dialog, when the TextMate bundle support plugin is enabled.

The plugin is bundled with PhpStorm and is activated by default. If it is disabled, you can manually enable the plugin .

Item	Description
TestMate Bundles	This table of added bundles consists of the following columns: – Check box: If a check box to the left of the added bundle name is selected, the bundle provides highlighting in the files of the corresponding type. – Name of the TextMate Bundle – Bundle location: for each added TextMate bundle, its location is shown.
+	Click this button to locate the desired bundle using the Select Path dialog box. When added, the bundle appears in the table of TextMate Bundles.
–	Click this button to remove the selected bundle. The bundle in question is removed from the list, but is not physically deleted from the computer.
IDE Color Scheme - TextMate Color Scheme	Use this table to establish mapping between the various color schemes of PhpStorm and TextMate. Tip If you want to use a custom TextMate color scheme, you can import a TextMate bundle with schemes, and it will become visible in the list of TextMate color schemes after clicking Apply.

TODO

File | Settings | Editor | TODO for Windows and Linux

PhpStorm | Preferences | Editor | TODO for OS X

`Ctrl+Alt+S`

In this page, configure filters to maintain your lists of TODO items in the TODO tool window and define TODO patterns to be used in filters.

Patterns

In this area, create and manage the list of available TODO patterns.

Item	Description
Icon	This read-only field displays the icon that is assigned to the current pattern. (This icon appears in the TODO tool window as a marker for the corresponding TODO items.)
Case sensitive	This read-only check box indicates whether the current pattern is case sensitive or not. The status is changed in the Edit Pattern dialog box.
Pattern	This read-only field displays the regular expression that describes the TODO pattern. PhpStorm recognizes regular expressions in the source code against the specified patterns and displays them in the TODO tool window.
+ (`Alt+Insert`)	Use this icon or shortcut to open the Add Pattern dialog box, where you can create a new ToDo pattern by specifying a regular expression.
(`Enter`)	Use this icon or shortcut to open the Edit Pattern dialog box, where you can modify the selected pattern.
— (`Alt+Delete`)	Use this icon or shortcut to remove the selected pattern from the list.

Filters

In this area, manage the list of available filters.

Item	Description
Name	This read-only field shows a list of filter names.
Patterns	This read-only field shows the names of patterns included in the current filter. A filter can contain several patterns.
+ (`Alt+Insert`)	Use this icon or shortcut to open the Add Filter dialog box, where you can define a new filter.
(`Enter`)	Use this icon or shortcut to open the Edit Filter dialog box, where you can edit the settings for the selected filter.
— (`Alt+Delete`)	Use this icon or shortcut to delete the selected filter.

Add / Edit Pattern Dialog

File | Settings | TODO - Add/Edit Pattern for Windows and Linux

PhpStorm | Preferences | TODO - Add/Edit Pattern for OS X

`Ctrl+Alt+S`

Use this dialog box to define patterns that help track TODO items in your source code. Make sure the TODO items in the source code are inserted inside comments that are valid for the supported file types.

Item	Description
Pattern	In this text box, type the regular expression that describes the desired TODO pattern.
Icon	From this drop-down list, choose an icon for the pattern.
Case sensitive	Select this check-box to make the pattern case-sensitive.
Font type	Select the corresponding check box to have the icon text displayed in bold or italic.
Foreground	Select this check box to enable the palette and choose the foreground color.
Background	Select this check box to enable the palette and choose the background color.
Error Stripe Mark	Select this check box to enable the palette and choose the error stripe color.
Effects	Select this check box to enable effects (underscore, strikethrough, etc.) and choose the color for them from the palette.

Add / Edit Filter Dialog

File | Settings | TODO - Add/Edit Filter for Windows and Linux

PhpStorm | Preferences | TODO - Add/Edit Filter for OS X

Ctrl+Alt+S

Use this dialog box to define filters that help track TODO items in your source code.

Item	Description
Name	In this text box, specify the name of the filter.
Patterns	From the list of available patterns, choose the ones to be included in the filter by selecting the check boxes next to them.

Plugins

File | Settings | Plugins for Windows and Linux

PhpStorm | Preferences | Plugins for OS X

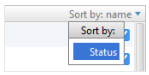
Ctrl+Alt+S

The Plugins page shows the list of installed plugins . Use the check boxes next to plugin names to enable or disable them.

Other controls on this page let you sort and filter the plugin list, update and uninstall repository plugins, access plugin repositories , and also install the plugins available locally.

- Main controls
- Context menu commands
- Colors for plugin statuses

Main controls

Item	Description
	Type the text to be found. As you type, the list of plugins changes: only the plugins whose names and descriptions contain the specified text are shown. To bring the plugin list back to its initial state, delete the text in the search box or click . To access the list of memorized search strings, click .
Show	Use this list to specify which plugins should be shown: − All plugins − Enabled − Disabled − Bundled − Custom (these are the plugins that are not bundled with the IDE).
	If you want to add sorting by plugin status, click and select Status. As a result, disabled plugins will be displayed at the end of the list. This option is also available from the context menu.
Install JetBrains plugin	Click this button to open the Browse JetBrains Plugins Dialog to download and install plugins from the JetBrains repository.
Browse repositories	Click this button to open the Browse Repositories dialog to work with plugin repositories (to download and install repository plugins, manage enterprise plugin repositories, etc.).
Install plugin from disk	Click this button to install a plugin available locally. Select the file that implements the required plugin in the dialog that opens .

Context menu commands

Command	Description
Reload List of Plugins	Use this command to check if newer versions of installed plugins are available (see Colors for plugin statuses).
Sort by | Status	Use this command to sort plugins by their status. When this option is turned on, disabled plugins are shown at the end of the list, after the enabled plugins.
Update Plugin	For repository plugins: use this command to download and install a newer version of the selected plugin (if available). See Colors for plugin statuses .
Uninstall	For plugins that are not bundled with the IDE: use this command to uninstall the selected plugin. Alternatively, the Uninstall button in the plugin description area can be used.

Colors for plugin statuses

The names of plugins are shown in different colors depending on their status.

Color	Plugin status
Black	"Normal" plugin status. For a repository plugin: the plugin version is up-to-date.
Red	One of the following: − The plugin is incompatible with the installed version of PhpStorm. − The plugin depends on another plugin which is disabled.
Blue	For a repository plugin: a newer version of the plugin is available.
Green	For a repository plugin: the plugin has been downloaded and installed, but has not been activated yet (PhpStorm needs to be restarted).
Gray	The plugin has been uninstalled, but the changes have not taken effect yet (PhpStorm needs to be restarted).

Browse Repositories Dialog

This dialog opens when you click the Browse repositories button on the Plugins page .

This dialog shows a list of repository plugins .

You can sort and filter the plugins list, download and install plugins, manage enterprise plugin repositories , and configure HTTP proxy settings.

- Main controls
- Context menu commands

Main controls

Item	Description
script	Type the search string. As you type, the list of plugins changes: only the plugins whose names and descriptions contain the specified string are displayed. If you want PhpStorm to remember the search string, press Enter. To bring the plugin list back to its initial state, delete the text in the search box or click . To access the list of memorized search strings, click .
	Use this icon to update the list of plugins and their statuses.
Repository: All	If you have enterprise plugin repositories configured, there is a list that provides repository-based filtering. You can select to see the contents of all repositories o,r only the selected one.
Category: All	This list provides category-based filtering. You can select to see all plugins, or the plugins that belong to the selected category.
Sort by	Click and select a category.
Install plugin	Click the Install plugin button in the description area to download and install the selected plugin.
HTTP Proxy Settings	If you access the Internet via an HTTP proxy, click this button and specify the HTTP proxy settings .
Manage repositories	Click this button to create or edit the list of enterprise plugin repositories in the Custom Plugin Repositories dialog .

Context menu commands

Command	Description
Reload List of Plugins	Use this command to update the list of plugins and their statuses.
Sort by | <Category>	Use this command to enable or cancel sorting. The following categories are available: - Status - Downloads - Rating - Last Updated
Download and Install	Use this command to download and install the selected plugin.

Browse JetBrains Plugins dialog

This dialog opens when you click the Install JetBrains plugin button on the Plugins page .

This dialog shows a list of the JetBrains repository plugins .

You can sort and filter the list, download and install plugins, and configure HTTP proxy settings.

- Main controls
- Context menu commands

Main controls

Item	Description
script	Type the search string. As you type, the list of plugins changes: only the plugins whose names and descriptions contain the specified string are displayed. If you want PhpStorm to remember the search string, press Enter . To bring the plugin list back to its initial state, delete the text in the search box or click . To access the list of memorized search strings, click .
	Click this icon to update the list of plugins and their statuses.
Category: All	This drop-down list provides category-based filtering. You can select to see all plugins, or only the plugins that belong to a the selected category.
Sort by	Click and select a category.
Install plugin	Click the Install plugin button in the description area to download and install the selected plugin.
HTTP Proxy Settings	If you access the Internet via an HTTP proxy, click this button and specify the HTTP proxy settings .

Context menu commands

Command	Description
Reload List of Plugins	Use this command to update the list of plugins and their statuses.
Sort by | <Category>	Use this command to enable or cancel sorting. The following categories are available: - Status - Downloads - Rating - Last Updated
Download and Install	Use this command to download and install the selected plugin.

Custom Plugin Repositories Dialog

This dialog opens when you click the Manage repositories button in the Browse Repositories dialog .

Use this dialog to manage the list of enterprise plugin repositories . The repositories are identified by their URLs .

Icon	Shortcut	Description
+	Alt+Insert	Use this icon or shortcut to add a new repository to the list. Specify the URL of the repository in the dialog that opens. Use the Check Now button to make sure that the specified URL is correct: PhpStorm will try to connect to the repository.
	Enter	Use this icon or shortcut to edit the selected URL.
−	Alt+Delete	Use this icon or shortcut to remove the selected URL from the list.

Version Control

File | Settings | Version Control for Windows and Linux

PhpStorm | Preferences | Version Control for OS X

Ctrl+Alt+S

The settings under this node allow configuring integration with different version control systems.

Common settings that are applied to the project files regardless of which version control system is used:

- Confirmation
- Background
- Ignored Files
- Issue Navigation
- Changelist Conflicts

The settings for configuring integration with a specific version control system are located under the following nodes:

- GitHub
- CVS
- Git
- Mercurial
- Perforce
- Subversion
- TFS

Specify which version control systems will be used for specific directories, or the entire project.

Item	Description
Directory	This field shows the path to project directories or the project root(s). For projects with Git or Mercurial integration enabled, PhpStorm scans project directories to check if there are Git /Mercurial repositories that are not controlled by the IDE. If such repositories are found, they are listed here under Unregistered roots and are marked grey. To add an unregistered root, select it in the list and click the Add button +. PhpStorm also checks if registered roots are valid, i.e. that a Git /Mercurial repository exists at the specified path. If invalid repositories are detected, they are marked with red.
VCS	Select a version control system for the specified directory. The list only displays the version control systems for which the corresponding plugins are enabled (See Managing Plugins).
...	Click this button to open the Version Control Configurations dialog and update the configuration settings for the selected VCS.
+	Click this button to add a directory mapping to the list. The Add VCS Directory Mapping dialog box opens where you can specify the required directory, select a VCS for it, and open the Version Control Configurations dialog box to configure the specified VCS, if necessary.
	Click this button to edit the selected directory mapping. The Edit VCS Directory Mapping dialog box opens where you can update the selected mapping and configure the specified VCS, if necessary.
−	Click this button to remove the selected directory mapping from the list.
Limit history to	Select this check box to specify the number of history rows. If this check box is selected, the text box of history depth, and the spin box become enabled.
Show directories with changed descendants	If this check box is selected, the directories that contain changes, are color-marked. The colors are configurable in the Colors and Fonts pages of the Editor settings (File Status - Have immediate changed children, Have changed descendants).
Store on shelf base revision texts of files under DVCS	Select this check box to have PhpStorm always shelve the base revisions of files that are under Git or Mercurial version control. By default, PhpStorm always "remembers" the last commit hash. However, this information is not sufficient if the history has been changed since the last commit as a result of running the Rebase operation. In this case, having a copy of the base revision may help. This check box is relevant only for integration with distributed version control systems, such as Git and Mercurial .
Show changed in last <number> days	Select this check box to have color indication of file status applied during stacktrace analysis and debugging . The names of the files that have been changed within a certain period will be highlighted accordingly. Specify the number of days.
Filter Update Project information by scope	If this option is enabled and a scope is selected, the files that belong to this scope will be marked in bold in the Update Project Info tab of the Version Control Tool Window . If you click the Filter button in the toolbar in the Update Project Info tab, the files will be filtered by scope, i.e. only the files that belong to the selected scope will be displayed. Click the Manage Scopes link to open the Scopes settings dialog and configure a scope.
Commit message right margin (columns)	In this field, specify the number of symbols that can fit into the right margin of the Commit Changes dialog. Select the Wrap when typing reaches right margin option if you want the text to be transferred to the next line when the maximum number of characters has been reached.
Show unversioned files in Commit dialog	Select this option to see newly added files that have not been added to version control yet under the Unversioned Files node in the Commit Changes dialog .

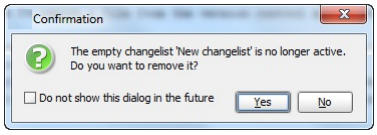
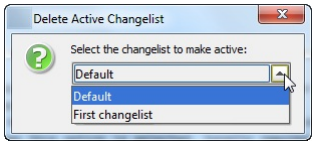
Confirmation

File | Settings | Version Control | Confirmation for Windows and Linux

PhpStorm | Preferences | Version Control | Confirmation for OS X

Ctrl+Alt+S

In this page, specify whether you want PhpStorm to ask you for confirmation before performing specific version control related actions.

Item	Description
When files are created	In this section, specify whether and how to put a file created in PhpStorm under version control. The following options are available: – Show options before adding to version control: When this option is selected, newly created files are put under version control after you specify the version control options in the dialog box that opens. – Add silently: When this option is selected, newly created files are automatically put under version control without displaying any messages. – Do not add: When this option is selected, newly created files remain unversioned and you can put them under version control later.
When files are deleted	In this section, specify whether and how to remove a file from the version control system when the file is removed in PhpStorm. The following options are available: – Show options before removing from version control: When this option is selected, locally removed files are also removed from the specified VCS but first a dialog box for selecting version control options is displayed. – Remove silently: When this option is selected, all locally removed files are removed from the specified VCS without asking for confirmation. – Do not remove: When this option is selected, locally removed files remain under version control.
When empty changelist becomes inactive	In this section, specify PhpStorm behavior on deleting an empty changelist. – Show options before removing: When this option is selected, PhpStorm asks for confirmation before removing an empty changelist that has lost its active status:  If you choose to delete such a changelist, PhpStorm suggests to choose another changelist to be marked as active.  – Remove silently: When this option is selected, PhpStorm automatically deletes empty changelists that become inactive, except for the Default changelist. – Do not remove: When this option is selected, empty changelist is not deleted on loosing its active status.
Display Option dialogs when these commands are invoked	In this area, specify whether you want PhpStorm to ask you for confirmation when invoking commands, specified below. To enable showing a prompt before performing an action, select the relevant check boxes. Tip – Availability of commands depends on the particular version control system. – Use controls in this area to restore the original settings, if you have suppressed showing operation-specific option dialog boxes by selecting the Do not show this dialog in the future check box.
Show Clear Read-Only Status dialog box	Select this check box to have PhpStorm explicitly require cancellation of the read-only status when you open a file in the editor and try to modify it. The read-only status can be cleared in two ways: – Using the current VCS: the file is added to a changelist. – Using a file system: the file is not added to the changelist. This option is relevant for those version control systems that separate check-out from opening for editing.
Suggest to move uncommitted changes to another changelist	When this option is selected, on committing a changelist to the repository with some files excluded from the commit, PhpStorm prompts to move these files to another changelist.
Force non-empty check-in comments	Select this check box to suppress committing changes without supplying them with corresponding comments.
Clear initial commit message	If this check box is selected, the previous check-in comment is cleared from the Comment area.
Show patch in explorer after creation	Use this drop-down list to define the PhpStorm behavior when a patch is created . The available options are: – Yes: if this option is selected, native file manager always opens to show the patch file. – No: if this option is selected, the native file manager will not open. – Ask: if this option is selected, PhpStorm will display a dialog box informing about successful patch creation, and a suggestion to locate the patch file in the native file manager.
Create changelist on failed commit	Use this drop-down list to define the PhpStorm behavior in case of failed commits. The available options are: – Yes: if this option is selected, the Failed commit changelist will be automatically created, and the respective files will be moved to this changelist. – No: if this option is selected, the files that failed to commit will remain in their original changelist. – Ask: if this option is selected, on failed commit PhpStorm will display a dialog box asking whether the files should be moved to another changelist, or remain in the original one. See the section Resolving Conflicts .

Background

File | Settings | Version Control | Background for Windows and Linux

PhpStorm | Preferences | Version Control | Background for OS X

Ctrl+Alt+S

In this page, enable performing specific version control related operations in the background without any activities from your side.

Item	Description
Background Operations	In this area, specify the version control related operations to be performed in the background. To enable running an operation in the background, select the relevant check box: − Perform update from VCS in background − Perform commit to VCS in background − Perform checkout from VCS in background − Perform Edit/Checkout in background − Perform Add/Remove in background − Perform revert in background
Changelists to cache initially	Use this spin box to specify the number of changelists to be stored in the cache.
Refresh changes every ... minutes	Use this spin box to specify how often the VCS should check for new changes and refresh the cache. The spin box is only enabled when the Enable background processes check box is selected.
"Changed on server" conflicts	In this area, specify whether you want PhpStorm to check whether a checked out files have been updated by someone else. To enable background synchronization with the server, select the Check every ... minutes. In the spin box, specify how often you want synchronization to take place. This area is only enabled when the Enable background processes check box is selected.

Ignored Files

File | Settings | Version Control | Ignored Files for Windows and Linux

PhpStorm | Preferences | Version Control | Ignored Files for OS X

`Ctrl+Alt+S`

Use this page to create a list of files and directories that you do not want to put under version control. These can be file names associated with VCS administration, backup files, and any other artifacts that you want to remain unversioned. You can also specify patterns that define files to ignore.

Item	Keyboard shortcut	Description
+	`Alt+Insert`	Use this icon or shortcut to add an item to the list. The Ignore Unversioned Files dialog box opens where you can type an exact path to a file or directory to be ignored or specify a pattern that defines the names of files and directories to be ignored.
	`Enter`	Use this icon or shortcut to edit the selected path or pattern in the Ignore Unversioned Files dialog box.
−	`Alt+Delete`	Use this icon or shortcut to remove the selected path or pattern from the list.

Patterns

Two characters are used as the wildcards:

- `*` : this character matches anything.
- `?` : this character matches a single character.

For example, `*.iml` will ignore all the files with the extension `iml` ; `*.?ml` will ignore all the files with the extension ending with `ml` .

Ignore Unversioned Files

File | Settings | Version Control | Ignored Files for Windows and Linux

PhpStorm | Preferences | Version Control | Ignored Files for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Add + or Edit button in the Ignored Files page.

Use this dialog to configure rules that define which files and folders should be ignored by version control systems. The files you want to ignore can be appointed explicitly by their names of through name patterns with wildcards. To ignore a directory, you need to specify the full path to it relative to the project root.

Select the relevant option and fill in the text box next to it.

Item	Description
Ignore specified file	In this text box, specify the name of the file to be ignored. Do one of the following: – Type the file name relative to the project root, for example, `my_folder/my_subfolder1/my_subfolder2/my_file`. – Click the Browse button ... and select the desired file in the Select File to Ignore dialog box.
Ignore all files under	In this text box, specify the name of the directory to be ignored. Do one of the following: – Type the directory name relative to the project root, for example, `my_folder/my_subfolder1/`. – Click the Browse button ... and select the desired folder in the Select Directory to Ignore dialog box. The rule is applied recursively to all the subdirectories of the specified directory. If a directory has several subdirectories and you want only one of them ignored, specify the required directory explicitly, for example, `my_folder/my_subfolder1/my_subfolder2/`.
Ignore all files matching	In this text box, specify a pattern that defines the names of files to ignore. The rule is applied to all the directories under the project root. Using wildcards in combination with slashes (`/`) to restrict the scope to a certain directory is not supported.

Issue Navigation

File | Settings | Version Control | Issue Navigation for Windows and Linux

PhpStorm | Preferences | Version Control | Issue Navigation for OS X

`Ctrl+Alt+S`

Use this dialog to create a list of the so-called issue navigation patterns. An issue navigation pattern maps an issue ID pattern in commit messages with the URL addresses of the referenced issues. This enables you to navigate from committed changes to issues related to these changes. As soon as PhpStorm encounters a match to the issue ID pattern in a commit message, the match is displayed as a link in the Version Control tool window . If you mention several issues, all of them will be displayed as links. Clicking such link opens the matching issue in the default browser.

Item	Description
Issue	This read-only field shows the issue pattern.
Link	This read-only field shows the link to navigate from the issue pattern in the current row to the issue in the bug tracking system.
+	Click this button to create a new issue navigation pattern and link. The Add Issue Navigation Link dialog box opens where you can specify: − A regular expression to define the issue ID. − A regular expression to define the navigation link to the issue. Tip Refer to Regular Expressions Syntax Reference for details on using special characters in regular expressions.
	Click this button to create a new JIRA pattern. The Create JIRA Issue Navigation Pattern dialog box is opened where you can specify the URL to your JIRA installation. The regular expression that defines the pattern is added automatically.
	Click this button to create a new pattern for YouTrack . In the dialog box that opens, specify the URL to your YouTrack installation. The regular expression that defines the pattern is added automatically.
	Click this button to update the selected issue navigation link.
−	Click this button to remove the selected issue navigation link from the list.

Example

The example below shows how PhpStorm applies the abovementioned rules to detect a reference to an issue in a commit message and compose a link to it in the issue tracking system.

Issue ID pattern

The regular expression that defines the format in which issues are referenced in commit messages.

```
[A-Z]+\-\d+
```

This regular expressions matches all character strings that consist of two substrings separated by an n-dash character:

1. Substring 1: An unlimited number of upper case alphabetic characters.
2. Substring 2: An unlimited number of digital characters.

Issue link pattern

A combination of the URL address of your issue tracking system and a regular expression that identifies issues in it.

```
http://mytracker/issue/$0
```

Here `$0` indicates a back reference to the entire match. This means that as soon as PhpStorm detects a match in a commit message, it is added to the URL address of the tracker as is.

Matching issue ID

PhpStorm detects the following reference to an issue in the commit message of interest:

```
MYPROJECT-110
```

Composed issue link

In accordance with the above issue navigation pattern, the detected matching reference is added to the URL of the tracker as is, so the link to the referenced issue is composed as follows:

```
http://mytracker/issue/MYPROJECT-110
```

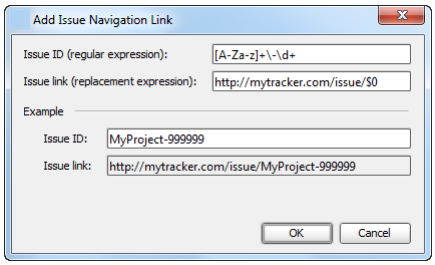
Add Issue Navigation Link Dialog

File | Settings | Version Control | Issue Navigation - Add for Windows and Linux

PhpStorm | Preferences | Version Control | Issue Navigation - Add foc OS X

Use this dialog box to create an issue pattern and navigation link to a bug tracking system.

Item	Description
Issue ID (regular expression)	In this field, type a regular expression that will be converted to the a specific issue id.
Issue link (replacement expression)	In this field, type a regular expression that will be converted to a navigation link to the issue.
Example	In this section, type some specific issue id to make sure it matches the specified pattern:



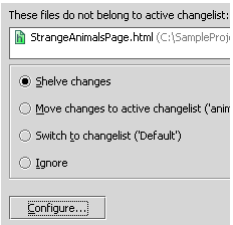
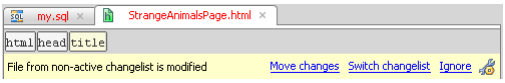
Changelist Conflicts

File | Settings | Version Control | Changelist Conflicts for Windows and Linux

PhpStorm | Preferences | Version Control | Changelist Conflicts for OS X

`Ctrl+Alt+S`

Use this page to configure protection of files belonging to inactive changelists, from occasional conflicts.

Item	Description
Enable changelist conflict tracking	If this check box is selected, PhpStorm makes it possible to protect files in inactive changelists. Such protection can be performed on the various levels enabled by the options listed below.
Show conflict resolving dialog	If this check box is selected, PhpStorm shows Resolve Changelist Conflict dialog box on an attempt to modify a file.  You need to define the way to resolve a conflict, before you proceed with changes. The possible ways to resolve a conflict are as follows: – Shelve changes. – Move file to the active changelist. – Switch changelists to make the current changelist active. – Ignore conflict. In this case, the file in question will be added to the list of files with ignored conflicts.
Highlight files with conflicts	If this check box is selected, PhpStorm shows a yellow stripe on top of a file from an inactive changelist, when such file has been modified.  In this stripe, you can choose one of the possible ways to resolve conflict: – Move file to the active changelist. – Switch changelists. – Ignore conflict. Names of the files belonging to inactive changelists are shown in red font in the editor tabs and in the Project view.
Highlight files from non-active changelists	If this check box is selected, names of the files belonging to inactive changelists are shown in light-blue font in the editor tabs and in the Project view.
Files with ignored conflicts	This area shows the list of files, for which Ignore option has been selected in the Resolve Changelist Conflict dialog box, or in the editor stripe.
Clear	Click this button to remove all files from the list of files with ignored conflicts.

GitHub

File | Settings | Version Control | GitHub for Windows and Linux

PhpStorm | Preferences | Version Control | GitHub for OS X

Ctrl+Alt+S

Use this page to specify your GitHub remote storage account credentials or create a GitHub account if you do not have one yet.

Item	Description
Host	Specify the URL of your GitHub repository.
Auth Type	Use this drop-down list to select how you want to be authenticated on GitHub. The available options are: – Password. If this option is selected and you have two-factor authentication enabled in your GitHub account settings, you will be asked to enter an authentication code each time PhpStorm requires you to log in to your GitHub account. – Token (recommended by GitHub for authentication from third-party applications, as it does not require PhpStorm to remember your password).
Login	In this text box, type your GitHub logon name. This field is only available if Password is selected as authentication method above.
Password	In this text box, type your GitHub account password. This field is only available if Password is selected as authentication method above.
Test	Click this button to verify the credentials you have specified.
Token	Specify your personal access token. This field is only available if Token is selected as authentication method above.
Create API Token	Click this button if you do not have a personal API token yet. Specify your GitHub credentials in the Login to GitHub dialog that opens and click the Login button. The token will be generated automatically.
Sign up	Click this link to open the Sign up for GitHub page where you can create a GitHub account.
Connection timeout	Specify the time period to wait for connection to be established.

CVS

File | Settings | Version Control | CVS for Windows and Linux

PhpStorm | Preferences | Version Control | CVS for OS X

`Ctrl+Alt+S`

Use this page to specify the version control settings to be applied to the directories of your project that are under CVS control.

Item	Description
Updating	In this area, define the PhpStorm behavior when merge conflicts occur during update from the server. – Show dialog - select this option to have PhpStorm display a dialog box where you can examine, analyze, and resolve possible conflicts before updating. – Skip merging for all project or module files merged with conflicts - select this option to suppress updating files where conflicts occurred during merge. – Get latest repository versions silently - select this option to have your local files in question automatically updated with their latest repository versions.
Use read-only flag for not edited file	Select this check box to have the read-only status assigned to a file automatically after the check out, update, or commit operations.
Show CVS server output	Select this check box to have the server output of CVS commands displayed in the CVS Output tool window.
Default keyword substitution for text files	Use this drop-down list to specify the keyword expansion mode. CVS uses the keyword substitution mode of a file to differentiate binary files from ASCII files and to indicate what type of keyword substitution is applied when files are committed and checked out. The available options are: – keyword&value (`-kkv`) – keyword, value&locker (`-kkvl`) – keyword only (`-kk`) – original string (`-ko`) – binary (`-kb`) – value only (`-kv`)
Global Settings	Click this button to open the Global CVS Settings dialog box.

Git

File | Settings | Version Control | Git for Windows and Linux

PhpStorm | Preferences | Version Control | Git for OS X

`Ctrl+Alt+S`

Use this page to specify the version control settings that will be applied to the directories of your project that are under Git control.

Item	Description
Path to Git executable	In this text box, specify the path to the Git executable file. Type the path manually or click the Browse button and specify the path in the dialog that opens.
Test	Click this button to verify the path to the Git executable file.
SSH executable	Use this drop-down list to specify the SSH version to be used with Git. The available options are: – Built-in: select this option to use the implementation provided by PhpStorm. – Native: select this option to use native implementation. Note that on some platforms, the use of native ssh implementation may cause hang-up issues. You may need to configure a platform-specific `ssh-askpass` to receive prompts for passwords.
Control repositories synchronously	This option only becomes available if you have a multirooted project, i.e. there are several Git repositories within a single project. Select this option if you want branch operations (such as `checkout`, `merge`, etc.) to be applied synchronously to all repositories within your project.
Commit automatically on cherry-pick	When you cherry pick a specific commit, the Commit Changes dialog is displayed. If the Commit automatically on cherry-pick option is selected, the selected commit is submitted silently on clicking the cherry-pick button without displaying the Commit Changes dialog.
Warn if CRLF line separators are about to be committed	Select this option to enable smart handling of `LF` and `CRLF` line separators. PhpStorm will analyze your configuration, warn you if you are about to commit CRLF into the repository, and suggest changing the `core.autocrlf` setting to `true` or `input` depending on your operating system. Note: Note that this setting is not applied to files where you have set any related Git attributes. In this case, PhpStorm assumes that you clearly understand what you are doing and excludes such files from analysis. If this option is deselected, you will have to fix issues with line endings manually using the Difference Viewer dialog.
Warn when committing in detached HEAD or during rebase	Select this option if you want PhpStorm to display a warning when a commit is performed from a detached head or on rebase, as this may cause issues and code loss.
Update method	Use this drop-down list to choose the strategy to synchronize your local repository with the remote storage. The selected method will be used when the `push` operation is rejected (if the Auto-updated if push of the current branch was rejected option is enabled), or when you invoke the Update Project operation. The following options are available: – Merge: choose this option to have the merge strategy applied. The result is identical with that of running `git fetch ; git merge` or `git pull --no-rebase`. – Rebase: choose this option to have the rebase strategy applied. The result is identical with that of running `git fetch ; git rebase` or `git pull --rebase`. – Branch Default: choose this option to have the default command for the branch applied. The default command is specified in the `branch.<name>` section of the `.git/config` configuration file.
Auto-update if push of the current branch was rejected	Select this checkbox if you want the current branch to be updated automatically if the `push` operation from the current branch to its tracked branch is rejected. If this option is deselected, PhpStorm will display the Push Rejected dialog when pushing a branch is rejected because your local repository and the remote storage are not synchronized. Note: Note the following: – If you have never seen the Push Rejected dialog box before and you are enabling the check box initially, PhpStorm will update the conflicting local branch silently by means of the `merge` operation. – If you have already encountered the Push Rejected dialog box and selected the Remember the update method choice... option, PhpStorm saves your last choice (`rebase` or `merge`) and will apply it to update the conflicting local branch silently. Accordingly, to change the "remembered" setting, clear the check box, access the Push Rejected dialog box, select the Auto-update if push ... rejected option, and invoke another update strategy.
Allow force push	If this check box is selected, the Force push option is added to the Git Push dialog (as a drop-down option on the Push button).
Protected branches	If you have selected the Allow force push option, but want to disable it for certain branches, list them here (this is a team-shared parameter that is stored in `.idea/vcs.xml`). You can list several branches separated by a colon, or supply branch patterns as the input is treated as a list of regular expressions.

Mercurial

File | Settings | Version Control | Mercurial for Windows and Linux

PhpStorm | Preferences | Version Control | Mercurial for OS X

`Ctrl+Alt+S`

Use this page to specify the version control settings to be applied to the directories of your project that are under Mercurial control.

Item	Description
Path to hg executable	Specify the location of the Mercurial executable file. Enter the path manually, or click the Browse button and select the path in the dialog that opens. If you followed the standard installation procedure, the default location is `/opt/local/bin` or `/usr/local/bin` for Linux and OS X and `/Program Files/TortoiseHG` for Windows. It is recommended that you add the path to the Mercurial executable file to the `PATH` variable. In this case, you can specify only the executable name, the full path to the executable location is not required. For more information about environment variables, see Path Variables .
Test	Click this button to verify the path to the Mercurial executable.
Control repositories synchronously	This option only becomes available if you have a multirooted project, i.e. there are several Mercurial repositories within a single project. Select this option if you want branch operations (such as `checkout`, `merge`, etc.) to be applied synchronously to all repositories within your project.
Check for incoming and outgoing changesets	Select this option if you want PhpStorm to detect incoming and outgoing changes in the background mode. PhpStorm will automatically request the server for incoming and outgoing changesets every 5 minutes.
Ignore whitespace differences in annotations	Select this check box if you want white spaces to be ignored when annotating, and, thus, get more meaningful annotations and cast out senseless ones.

Subversion

File | Settings | Version Control | Subversion for Windows and Linux

PhpStorm | Preferences | Version Control | Subversion for OS X

`Ctrl+Alt+S`

Use this page to specify the version control settings to be applied to the directories of your project that are under Subversion control

In this section:

- General tab
- Presentation tab
- Network tab
- SSH Settings

General tab

Use this tab to configure the general Subversion integration settings.

Item	Description
Use command line client	Select this option if you want to use the command line svn client. Enter the name of the executable file, or click the Browse button ... and select the path in the dialog that opens.
Enable interactive mode	Select this option if you want PhpStorm to emulate the behavior when Subversion commands are executed directly from the terminal in the interactive mode (dialogs will pop up where you can input credentials). This is required to handle password/passphrase prompts for svn+ssh repositories, and trust invalid server certificates for https repositories.
Use system default Subversion configuration directory	Select this check box to store Subversion configuration files in the system default location.
Subversion configuration directory	In this text box, specify the Subversion configuration directory if you do not want to use the default one. Enter the path manually, or click the Browse button ... and select the path in the dialog that opens. This option is only available when the Use system default Subversion configuration directory option is deselected.
Update administrative information only in changed subtrees	This option only applies to working copies older than SVN 1.7 managed by SVNKit. During synchronization with the server (update), SVN locks your working copy one subtree after another by creating empty `lock` files in the corresponding administrative `.svn` directories. After that, SVN starts comparing file hashes to detect which local files need to be synchronized. When this option is selected, SVN first checks if any files from a subtree have been modified on the server, and locks this subtree (i.e. creates a `.svn/lock` file) only if such files are detected. This approach improves performance but may cause concurrency issues, for example, with antiviral software.
Clear Auth Cache	Click this button to delete all stored credentials for the `http`, `svn` and `svn+ssh` protocols from the authentication cache.

Presentation tab

Use this tab to configure the data presentation settings.

Item	Description
Check svn:mergeinfo in target subtree when preparing for merge	Select this check box if you want PhpStorm to check the merge tracking information for the target branch before merging to prevent duplicates.
Maximum number of revisions to look back in annotations	Select this check box to limit the number of revisions to look back at when calculating annotations, and specify the number of revisions.
Show merge source in history and annotations	Select this check box if you want merge sources to be visible in annotations and file history.
Ignore whitespace differences in annotations	Select this check box if you want white spaces to be ignored when annotating, and, thus, get more meaningful annotations and cast out senseless ones.

Network tab

Use this tab to configure the connection settings.

Item	Description
Use PhpStorm general proxy settings as default for Subversion	Select this check box if you want Subversion to use the default PhpStorm proxy settings.
HTTP timeout	Specify the number of seconds to wait for HTTP connection to be established.
SSH connection timeout	Specify the number of seconds to wait for SSH connection to be established.
SSH read timeout	Specify the number of seconds to wait for response.
SSL protocols	In this area, select which SSL protocol you want to use. The available options are: – All – SSLv3 – TLSv1
Edit Network Options	Click this button to change Subversion runtime configuration file in the Edit Subversion options related to the network layers dialog .

SSH Settings

Use this tab to configure the settings used to connect to an SVN server via a tunneling SSH protocol.

Item	Description
SSH executable	Specify the path to the SSH client. Enter the name of the executable file, or click the Browse button ... and select the path in the dialog that opens. If not specified, `'ssh'` is used by default. This field is only available if the Password or the Private key option is selected.
User name	Specify the user name for SSH connection. If the user name is explicitly specified in the repository URL, this value will be used and this setting will be ignored. This field is only available if the Password or the Private key option is selected.
Port	If your server is listening on a non-standard port (22 for svn+ssh://), modify the default value. This field is only available if the Password or the Private key option is selected.
Password	Select this option if you want to use a password for SSH authentication.
Private key	Select this option if you want to use a private key for SSH authentication.
Path	Specify the path to the private key. Enter the path manually, or click the Browse button ... and select the path in the dialog that opens.
Subversion config	Select this option if you want to use the default settings stored in Subversion configuration for SSH connection.
SSH tunnel	This field displays the SSH tunnel settings stored in Subversion configuration. You can modify the value and click the Update button to write this value to the Subversion configuration.
Update	Click this button to check the Subversion configuration and update the value if necessary, or to write the value you have entered to the Subversion configuration.
SVN_SSH	This field displays the environment variable that can be used in the tunnel configuration (by default, `SVN_SSH`) and is stored in Subversion configuration.

Edit Subversion Options Related to Network Layers Dialog

File | Settings | Version Control | Subversion for Widnwos and Unix

PhpStorm | Preferences | Version Control | Subversion for OS X

The dialog box opens when you click the Edit Network Options button on the Subversion page of the Settings dialog box. In this dialog box, specify the Subversion network settings stored in the servers Subversion runtime configuration file.

The dialog box contains two tabs:

- System file - this tab displays the default network configuration settings specified by the system administrator.
- User file - in this tab, customize the default network configuration settings.

The dialog box consists of two panes:

- On the left-hand pane, add, edit, and remove configuration profiles. Network configuration settings are arranged into profiles of two types:
  - Group - settings from such profile apply to a specific group, defined by a glob pattern.
  - Global - settings from this profile are applied to all servers that do not match any glob pattern.
- On the right-hand pane, specify the settings for the selected configuration profile.

Toolbar Options

Item	Tooltip and shortcut	Description
+	Add Insert	Click this button to have a new configuration profile added to the list.
−	Delete Delete	Click this button to remove the selected profile from the list.
	Copy Ctrl+C	Click this button to have a copy of the selected profile created.

HTTP Proxy Settings

Item	Description
URL Patterns	In this text box, type the patterns that define the URL addresses of repositories to be accessed via proxy. Use commas to separate patterns.
Exceptions	In this text box, type the patterns that define the URL addresses of repositories to be accessed directly, without using proxy. Use commas to separate patterns.
Server	In this text box, specify the name or IP address of the proxy server to use.
Port	In this text box, specify the port number the proxy server listens to.
Connection timeout	In this text box, specify the time period in seconds after when the Subversion client cancels its attempts to establish connection to the server.
User	In this text box, type the user name or login to authenticate at the specified proxy server.
Password	In this text box, type the password that matches the specified login or user name.

SSL Settings

Item	Description
Comma separated paths to CAs certificate files	In this text box, specify the paths to files that contain certificates of the Certificate Authority (CAs) files that are accepted by the Subversion client when accessing the repository.
SSL client certificate file	In this text box, specify the location of the SSL client certificate file. Type the path to the file manually or click the Browse button ... and choose the location in the dialog that opens .
SSL client certificate passphrase	In this text box, type the SSL client certificate passphrase to use.
Trust default CAs	Select this check box to have the Subversion integration trust the set of default Certificate Authority files shipped with OpenSSL .

Repositories

Item	Description
Repositories	This list displays the URL addresses of the previously accessed repositories.
Test connection	Click this button to make sure that connection to the selected repository can be established successfully according to the settings specified in the dialog box. Tip When you click this button, PhpStorm displays the Authentication Required dialog box.

TFS

File | Settings | Version Control | TFS for Windows and Linux

PhpStorm | Preferences | Version Control | TFS for OS X

Ctrl+Alt+S

Use this page to specify the version control settings to be applied to those directories of your project that are under TFS control.

Item	Description
Servers and workspaces	In this area, configure access to workspaces and servers to use. – Manage: click this button to open the Manage TFS Servers and Workspaces dialog box where you can create a list of servers and workspaces you need to have access to. – Use PhpStorm HTTP Proxy settings for TFS: when this check box is selected, TFS servers are accessed through the Proxy server using the PhpStorm default Proxy settings .
Passwords	In this area, handle the passwords for accessing TFS servers and workspaces. – Reset Saved Passwords: click this button to discard the stored passwords.
Check-in policies compatibility	A check-in policy is a rule that is executed before every check-in to ensure that the selected changeset is OK to commit. Standard policies are stored on the server and are executed on the client machines. Custom policies are implemented as custom plugins to PhpStorm. The IDs of these plugins are stored on the server, while the policies themselves are applied locally. Therefore, to enable the use of a policy in a team, all the team members should install the corresponding plugin. Use the controls in this area to configure how PhpStorm should treat third party check-in policies. These settings are applied to all PhpStorm projects by default unless they are overridden for a specific project. – Evaluate Team Explorer policies: select this check box to have the Microsoft Team Explorer policy definitions installed and executed on the client machine. – Evaluate Teamprise policies: select this check box to have the Teamprise policy definitions installed and executed on the client machine. – Warn about not installed policies: select this check box to have warnings displayed in case the specified policy definition is not installed.

Manage TFS Servers and Workspaces

File | Settings | Version Control | TFS for Windows and Linux

PhpStorm | Preferences | Version Control | TFS for OS X

Ctrl+Alt+S

The dialog box opens when you click the Manage servers and workspaces button in the TFS dialog box. Use this dialog box to handle the list of TFS servers and workspaces you have access to.

Item	Description
Server/Workspace	This read-only field shows the URL addresses of TFS servers you have access to and workspaces available on these TFS servers.
Workspace comment	This read-only field shows the descriptions of workspaces on the servers you have access to.
Team Servers	Use the buttons in this area to manage the list of available servers and workspaces and configure access to them. – Add: click this button to open the Add Team Foundation Server dialog box where you can specify the parameters for establishing connection to a TFS server. TFS uses NTLM authentication , so native Windows applications (that is, Microsoft Team Explorer) authenticate silently with system credentials. PhpStorm users must always specify their username and password because of limitations posed by Java Runtime. – Remove: click this button to remove the selected server from the list. – Reload workspaces: click this button to have the list of available workspaces refreshed. – TFS Proxy: click this button to open the Set TFS proxy for server... dialog box where you can specify the parameters for accessing the selected server via Proxy. – Check-in Policies: click this button to open the Edit Check-in Policies dialog box where you can manage the list of check-in policies to be applied.
Workspaces	Use the buttons in this area manage the list of available workspaces and update the workspaces, when applicable. – Create: click this button to open the Create Workspace dialog box for creating a new workspace. – Edit: click this button to open the Edit Workspace dialog box for editing the selected workspace. – Delete: click this button to remove the selected workspace from the list.
Close	Click this button to save the settings, close the dialog box, and return to the TFS dialog box.

Add Team Foundation Server

File | Settings | Version Control | TFS for Windows and Linux

PhpStorm | Preferences | Version Control | TFS for OS X

Ctrl+Alt+S

The dialog box opens when you click the Add button in the Manage TFS Servers and Workspaces dialog box. Use this dialog box to specify and edit the parameters for establishing connection to TFS servers.

Item	Description
Address	In this text box, type the URL address of the TFS server you want to connect to.
Auth	In this field, specify the authentication protocol to access the server. TFS uses NTLM authentication , so native Windows applications (that is, Microsoft Team Explorer) authenticate silently with system credentials. PhpStorm users must always specify their username and password because of limitations posed by Java Runtime. To authenticate through OAuth (Windows Live ID), choose Alternate from the Auth drop-down list.
User name	In this text box, type your TFS user name. The field is available for the NTLM) and Alternate authentication types.
Domain	In this text box, type the name of the network domain where the TFS server is located. The field is available for the NTLM) authentication type.
Password	In this text box, type your TFS password. The field is available for the NTLM) and Alternate authentication types.
Store password	Select this check box to have PhpStorm remember the specified password.

Create Workspace

File | Settings | Version Control | TFS for Windows and Linux

PhpStorm | Preferences | Version Control | TFS for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Create button or select a workspace and click the Edit button in the Manage TFS Servers and Workspaces dialog box.

Use this dialog box to define a new workspace or update the existing one by selecting the necessary folders on the TFS server and mapping them to local folders.

Item	Description
Name	In this text box, specify the name of the new workspace.
Comment	In this text box, describe briefly what this workspace is intended for.
Server	This read-only field displays the URL address of the TFS server on which the new workspace will be created and which is selected in the Manage TFS Servers and Workspaces dialog box.
Owner	This read-only field displays your TFS user name.
Computer	This read-only field displays the name of your computer in the network domain.

Working Folders

In this area, define mappings between the necessary folders on the TFS server and local folders.

Tip You can map a folder including all its subfolders recursively or map each subfolder separately.

Item	Description
Status	From this drop-down list, select the status of a new mapping.
Local path	In this text box, specify the path to the local folder. Use the ... button, if necessary.
Server path	In this text box, specify the path to the corresponding folder on the server.
Add	Click this button to create a new mapping.
Remove	Click this button to remove the selected mapping from the list.

Edit Check-in Policies Dialog

File | Settings | Version Control | TFS for Windows and Linux

PhpStorm | Preferences | Version Control | TFS for OS X

Ctrl+Alt+S

The dialog box opens when you select an entry in the Server/Workspace list and click the Check-in Policies button in the Manage TFS Servers and Workspaces dialog box.

A check-in policy is a rule that is executed before every check-in to ensure that the selected changeset is OK to commit. Standard policies are stored on the server and are executed on the client machines.

Custom policies are implemented as custom plugins to PhpStorm. The IDs of these plugins are stored on the server, while the policies themselves are applied locally. Therefore, to enable the use of a policy in a team, all the team members should install the corresponding plugin.

Use this dialog box to manage the list of the custom project policies to be applied when checking in to the selected workspace and to override the default PhpStorm-wide policies for the project, if necessary.

In this section:

- Check-in Policies
- Compatibility

Check-in Policies

Item	Description
Team Project	From this drop-down list, select the name of the project to specify the policies for.
Policy Type	This read-only field shows the available policies.
Description	This read-only field shows brief descriptions of policies.
Enabled	When this check box is selected, the policy next to it is mandatory during check-in.
Add	Click this button to open the Add Check-in Policy dialog box where you can define a new check-in policy.
Edit	Click this button to open the Edit Check-in Policy dialog box where you can re-define the selected check-in policy.
Remove	Click this button to remove the selected check-in policy from the list.

Compatibility

Use the controls in this area to suppress applying the default PhpStorm-wide check-in policy settings to the current project.

Item	Description
Override default settings for team project <project name>	Select the check box to discard the default policy settings within the scope of the current project and re-define the settings by selecting or clearing the corresponding check boxes below. – Evaluate Team Explorer policies: select this check box to have the Microsoft Team Explorer policy definitions installed and executed on the client machine. – Evaluate Teamprise policies: select this check box to have the Teamprise policy definitions installed and executed on the client machine. – Warn about not installed policies: select this check box to have warnings displayed in case the specified policy definition is not installed.

Perforce

File | Settings | Version Control | Perforce for Windows and Linux

PhpStorm | Preferences | Version Control | Perforce for OS X

`Ctrl+Alt+S`

Use this page to specify the version control settings to be applied to those directories of your project that are under Perforce control

Item	Description
Perforce is online	Select this check box to work with Perforce in the online mode.
Switch to offline mode automatically if Perforce is unavailable	Select this check box to have PhpStorm automatically go offline as soon as Perforce becomes unavailable and display the corresponding message.
Use P4CONFIG or default connection	If this option is selected, `P4CONFIG` or the default Perforce connection are used to connect to the Perforce server. Using `P4CONFIG` makes it trivial to switch between Perforce settings for different projects, when necessary.
Use connection parameters	If this option is selected, the connection credentials (port, client, user name, and charset) are specified manually.
Port	In this text box, type the server and the port which your Perforce client will listen to. For the default Perforce server configuration, it looks like *perforce:1666*.
Client	In this text box, type the name of your Perforce workspace name.
User	In this text box, type your user name to authenticate to the server.
Charset	From this drop-down list, select the character set to be used.
Dump Perforce commands to <path>	Select this check box to have PhpStorm create a file `P4.output` and store the output of Perforce commands in it.
Use login authentication	When this check box is selected, Perforce requires a login to authenticate a user.
Test Connection	Click this button to check whether the specified settings ensure establishing connection to the Perforce server.
Path to P4 executable	In this text box, specify the path to the Perforce Command Line Client's executable file `P4`. Click the Browse button ... to open the Select Path - P4 Configuration dialog box and select the executable file in the directories tree.
Path to P4V executable	In this text box, specify the path to the Perforce Visual Client's executable file `P4V`. Click the Browse button ... to open the Select Path - P4 Configuration dialog box and select the executable file in the directories tree.
Show branching history ...	Select this check box to enable displaying the branch history of a specified file, including all file branch points, edits, and merges.
Show integrated changelists in committed changes	Select this check box to have PhpStorm point at committed changes that are also integrated to other changelists and provide information on the target changelists that received the content in question.
Server timeout	In this text box, specify the time period in seconds after when the Perforce client cancels its attempts to establish connection to the server.
Enable Perforce Jobs Support	When this check box is selected, user interface for attaching and detaching Perforce jobs to change lists is provided in the Version Control tool window and in the Commit Changes dialog box.

Directories

File | Settings | Directories for Windows and Linux

PhpStorm | Preferences | Directories for OS X

Ctrl+Alt+S

Use this dialog box to configure content roots of a project.

ItemTooltipDescription		
+	Add Content Root	Click this button and select a new content root folder in the dialog that opens .
	Remove Content Entry	Click this button to have the selected root removed from the list.
	Tests	Click this button to mark the selected folder as a test root.
	Sources	Click this button to mark the selected folder as the root for namespaces used in your project. Based on this setting, PhpStorm suggests you the proper folder name when you want to create a new namespace under another parent namespace during creation or moving a PHP class, that is, when you are actually creating or moving a PHP class to a non-existing namespace under another parent namespace. If no Sources folder is specified, you will have to type the proper folder manually. Appointing a Sources folder is not mandatory but this helps you keep your project structure in compliance with the PSR0 and PSR4 standards. See Configuring PHP Namespaces in a Project for details.
	Excluded	Click this button to mark the selected folder as excluded so PhpStorm ignores it during indexing, parsing, and code completion.
	Resource Root	Click this button to enable PhpStorm to complete relative paths to resources under the selected folder.
	Include	Click this button to have the exclusion mark removed from the selected excluded content root.

Build, Execution, Deployment

File | Settings | Build, Execution, Deployment for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment for OS X

When you select the Build, Execution, Deployment category in the left-hand pane, its main subcategories are listed in the right-hand part of the dialog.

- Debugger
- Deployment
- Docker
- Docker Registry
- Coverage

Debugger

File | Settings | Build, Execution, Deployment | Debugger for Windows and Linux

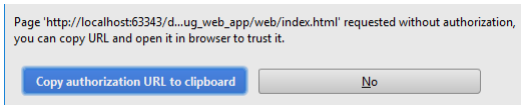
PhpStorm | Preferences | Build, Execution, Deployment | Debugger for OS X

Use this page to configure behavior of the Debugger and customize its view.

Common options

Item	Description
Focus application on breakpoint	If this check box is selected, on hitting a breakpoint, PhpStorm will show the location of this breakpoint in the editor and will attempt to bring its frame to the front.
Show debug window on breakpoint	Is this check box is selected, PhpStorm activates the Debug Tool Window on hitting a breakpoint.
Hide debug window on process termination	Automatically hide the Debug window when the debugged program terminates.
Scroll execution point to center	If this check box is selected, the line with the current execution point will be kept in the middle of the screen.

Built-in server

Item	Description
Port	Use this spin box to specify the number of the port where the built-in web server runs. The port number should be set to 1024 or higher. By default this port is set to the default PhpStorm port through which PhpStorm accepts connections from services (`63342`).
Can accept external connections	If this check box is selected, then the files launched on the built-in server running on the specified port, are accessible from a different computer. If this check box is not selected (by default), then only local connections are listened to.
Allow unsigned requests	For security reasons, any request to a page on the built-in server from outside PhpStorm is by default rejected and the following authorization pop-up window is displayed:  To access the requested page, click Copy authorization URL to clipboard and paste the generated token in the address bar of the browser. However this behaviour may be annoying, for example, it may block your debugging session if manual intervention is impossible. To suppress displaying the authorization pop-up window, select the Allow unsigned requests check box.

Data Views

File | Settings | Build, Execution, Deployment | Debugger | Data Views for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Debugger | Data Views for OS X

`Ctrl+Alt+S`

Use this page to manage the way data is displayed in the debugger.

On this page:

– Common debugger settings

– Editor

– JavaScript

Common debugger settings

Item	Description
Sort values alphabetically	Select this option to sort the values in the Variables pane of the Debug tool window .
Enable auto expressions in Variables view	Select this option if you want the PhpStorm debugger to automatically evaluate expressions and show the corresponding values in the Variables pane of the Debug tool window . The debugger analyzes the context near the breakpoint (the current statement, one statement before, and one after). It does so to find various expressions in the source code (if available) such as, for example, `myvar.myfield` . If such expressions don't contain explicit method invocations, the debugger evaluates them and shows the corresponding values in the Variables view.

Editor

Item	Description
Show values inline	Select this option to enable the Inline Debugging feature that allows viewing the values of variables right next to their usage in the editor.
Show value tooltip	Select this option to enable automatic display of tooltips for values. A tooltip in this context is a pop-up that provides an alternative, sometimes a more convenient presentation of values in the Variables pane of the Debug tool window . If this option is disabled, press `Alt` to display a value.
Value tooltips delay (ms)	Specify the delay (in milliseconds) between the moment when the mouse pointer hovers over an object in the Variables pane of the Debug tool window , and the moment when a tooltip with the object's value is displayed.
Show value tooltip on code selection	Select this option to enable tooltips that show the expression value when you select a code fragment in the editor.

JavaScript

Item	Description
Show the following properties for an object node	Select this check box if you want PhpStorm to show certain object node properties and configure a list of the properties to display. For each object node in the Variables pane, PhpStorm will display a label with the values of the listed properties.

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new property.
–	`Alt+Delete`	Click this button re remove the selected property.

Stepping

File | Settings | Build, Execution, Deployment | Debugger | Stepping for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Debugger | Stepping for OS X

`Ctrl+Alt+S`

Use this page to improve the debug stepping speed and specify the elements to be skipped while stepping.

Item	Description
JavaScript	
Do not step into library scripts	Select this check box to suppress stepping into library scripts while debugging.
Do not step into scripts	Select this check box to suppress stepping into certain scripts while debugging. Use the toolbar buttons to manage the list of scripts to be skipped.

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add a new script filter.
−	`Alt+Delete`	Click this button to delete the selected filter from the list.
	`Enter`	Click this button to edit the selected filter.
/		Use these buttons to arrange filters as required.
		Click this button to create a copy of the selected filter.

Live Edit

File | Settings | Build, Execution, Deployment | Debugger | - Live Edit for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Debugger | - Live Edit for OS X

`Ctrl+Alt+S`

The page is available only when the LiveEdit plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to enable and disable instant preview of HTML, CSS, and JavaScript files in the actual browser and configure the appearance of the preview.

Item	Description
Highlight current element in browser on caret change	When this check box is selected, the browser highlights the area that represents the code you are currently editing in PhpStorm.

Update	In this area, configure the way changes made to the code during a debugging session are applied. Note that an update will now be performed only if none of the modified files have any syntax errors.
	– Auto in (ms): Choose this option to have the changes applied automatically at certain time interval and specify the delay in ms in the text box. This policy is not available for the client-side code of Meteor applications.
	– Manual: Choose this option to apply the changes manually by clicking the Update <run configuration name> JavaScript button or the Update <run configuration name> button on the toolbar of the Debug tool window.
	It is recommended that you choose this option for debugging Meteor applications because applying changes to the client-side code is not supported.
	– Restart if hotswap fails:
	With changes in HTML, CSS, and JavaScript on the client side, the contents of a Web page in the browser are updated without reloading. For NodeJS or Meteor applications, PhpStorm first tries to update the application incorporating the changes without restarting the NodeJS server.
	  – Select this check box to have PhpStorm try restarting the server if the changes cannot be applied automatically. After the restart, PhpStorm brings you to the execution point where the problem took place.
	  If even with this option chosen automatic upload still fails, tou will have to restart the server manually by clicking the Rerun <run configuration name> button .
	  – When the check box is cleared, PhpStorm just displays a pop-up window informing you about the failure and suggesting you to restart the server manually.

Deployment

File | Settings | Build, Execution, Deployment | Deployment for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Deployment for OS X

`Ctrl+Alt+S`

In this section:

- Deployment
  - Basics
  - Toolbar and common options
- Deployment: Connection Tab
- Deployment: Mappings Tab
- Deployment: Excluded Paths Tab
- Add Server Dialog
- Options
- Advanced Options Dialog
- Files/Folders Default Permissions Dialog

Basics

On this page, create, edit, and delete server access configurations that give you control over interaction between PhpStorm and servers. Anytime you are going to use a server, you need to define a server access configurations, no matter whether your server is on a remote host or on your computer.

Among numerous ways to configure your development and production environments the most frequent ones are as follows:

- The Web server is installed on your computer. The sources are under the server document root (for example, `/htdocs`), and you do your development right on the server.
- The Web server is installed on your computer but the sources are stored in another folder. You do your development, then copy the sources to the server.
- The Web server is on another computer (remote host). Files on the server are available through the FTP/SFTP/FTPS protocol, through a network share, or a mounted drive.

Note that PhpStorm assumes that all development, debugging, and testing is done on your computer and then the code is deployed to a production environment. For detailed reasoning of this approach, see Working with Web Servers: Copying Files

Let's define the terms and their meaning in the context of synchronization between PhpStorm and servers.

- An in-place server is a server whose document root is the parent of the project root, either immediate or not. In other words, the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.
- A local server is a server that is running in a local or a mounted folder and whose document root is NOT the parent of the project root.
- A remote server is a server on another computer (remote host).
- The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.
- A local file/folder is any file or folder under the project root.
- A remote file/folder is any file or folder on the server, either local or remote.
  Suppose you have a project `C:/Projects/My_Project/` with a folder `C:/Projects/My_Project/My_Folder` and a local server with the document root in `C:/xampp/htdocs`. You upload the entire project tree to `C:/xampp/htdocs/My_Project`. In the terms of PhpStorm, the folder `C:/Projects/My_Project/My_Folder` is referred to as local and the folder `C:/xampp/htdocs/My_Project/My_Folder` is referred to as remote.
- Upload is copying data from the project TO the server, either local or remote.
- Download is copying data FROM the server to the project.

Synchronization with servers, uploading, downloading, and managing files on them are provided via the Remote Hosts Access bundled plugin, which is by default enabled. If the plugin is disabled, activate it in the Plugins page of the Settings dialog box. For details, see Enabling and Disabling Plugins .

Toolbar and common options

Use the toolbar buttons to manage the list of configurations.

The left-hand pane shows a list of all the server access configurations available in PhpStorm. When you select a configuration, the right-hand pane shows the configuration details.

Item	Tooltip and shortcut	Description
+	Add `Insert`	Click this button to open the Add Server dialog box and define a new configuration there.
−	Delete `Delete`	Click this button to remove the selected configuration from the list.
	Copy `Ctrl+D`	Click this button to copy the settings of the selected configuration.
	Use as Default	Click this button to have PhpStorm apply the settings of the selected configuration by default during automatic upload of changed files.

Deployment: Connection Tab

Use this tab to choose the way to access the Web server and specify the connection settings .

Item	Description
Name	The text box shows the configuration name specified in the Add Server dialog box. Edit the configuration name, if necessary.
Visible only for this project	Use this check box to configure the visibility of the server access configuration (deployment configuration). – Select the check box to restrict the use of the configuration to the current project. Such configurations cannot be reused outside the current project, they do not appear in the list of available configurations in other projects. For example, if this check box is selected in an SFTP configuration, you cannot use your SSH credentials from it when you configure a remote interpreter. – When the check box is cleared, the configuration is visible in all PhpStorm projects and the settings from, including SSH credentials, can be reused. See Configuring Node.js Interpreters and Configuring Remote PHP Interpreters for details.
Access type	From this drop-down list, choose the way to access the server. Use the Up and Down keyboard keys to scroll through the list of server configuration types. The available options are: – FTP: choose this option to have PhpStorm access the server via the FTP file transfer protocol . – SFTP: choose this option to have PhpStorm access the server via the SFTP file transfer protocol. – FTPS: choose this option to have PhpStorm access the server via the FTP file transfer protocol over SSL (the FTPS extension). – Local or mounted folder: choose this option if the Web server is running in a local or a mounted folder and its document root is NOT the parent of the project root. – In-place: choose this option if the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.

Upload/Download Project Files

In this area, specify the settings for accessing the server to upload and download files to and from.

The set of controls in the area depends on the chosen server access type.

Item	Description	Available for
Folder	In this field, specify the server configuration root. The server configuration root is the highest folder in the file tree on the server that can be accessed through the server configuration. The easiest way is to use the document root of your Web server as defined in the Web server configuration file. However you can appoint any other existing folder under the document root.	
FTP/FTPS/SFTP host	In this text box, specify the host name of the FTP/SFTP server to upload the files to.	
Port	In this text box, specify the port to use. The default values are: – 21 for FTP and FTPS – 22 for SFTP	
Root Path	In this text box, specify the server configuration root relative to your user home which was defined when you registered your account. This folder will be the highest one in the folder structure accessible through the current server configuration. Do one of the following: – Accept the default value `/` , which points at the user home folder on the server. – Type the path manually. – Click the Browse button ... and select the desired folder in the Choose Root Path dialog box that opens. – Click the Autodetect button and have PhpStorm detect the user home folder settings on the FTP/SFTP server and set up the root path according to them. The button is only enabled when you have specified your user name and password.	
Autodetect	Click this button to have PhpStorm detect the user home folder settings on the FTP/SFTP server and set up the root path according to them.	
User name	In this text box, type your user name for authentication to the server. Note The button is only enabled when you have specified your user credentials.	
Log in as anonymous	Select this check box to enable anonymous access to the server with your email address as password.	
Auth type	From this drop-down list, select the client authentication method. The available options are: – Password - select this option to use standard authentication through a password. – Key pair (OpenSSH) - select this option to use SSH authentication via a key pair. To apply this authentication method, you need to have your private key on the client machine and your public key on the remote server you connect to. PhpStorm supports private keys generated using the OpenSSH utility. See http://wiki.qnap.com/wiki/How_To_Set_Up_Authorized_Keys for details.	
Password	In this text box, type your password for authentication to the server.	
Private key file	In this text box, specify the location of your private key file.	
Passphrase	In this text box, specify your authentication passphrase.	
Save password	Select this check box to have PhpStorm remember the specified password.	
Save passphrase	Select this check box to have PhpStorm remember the specified passphrase.	
Test FTP/FTPS/SFTP connection	Click this button to check that the specified settings ensure successful connection via FTP/SFTP.	
Explicit	Choose this option to have the explicit (active) security applied. Immediately after establishing connection, the FTP client on your machine sends a command to the server to establish secure control	

	connection through the default FTP port.
Implicit	Choose this option to have the implicit (passive) security applied. In this case, security is provided automatically upon establishing connection to the server which appoints a separate port for secure connections.
Advanced options	Click this button to specify additional uploading settings in the Advanced Options dialog box that opens.
Web server root URL	In this text box, specify the URL address of the Web server root folder . Both the HTTP and the HTTPS protocols are supported. To access a server through HTTPS, you need to acquire a certificate file `<certificate_name>.cert` signed by a recognized authority and import this certificate in the truststore/keystore of the Oracle JRE (Java Runtime Environment) on which PhpStorm runs. Note that self-signed certificates are rejected as unsafe.

To import a certificate in Oracle JRE:

1. Open the embedded Terminal and type the following command:

```
<jre_home>/bin/keytool.exe -importcert -keystore <path to jre truststore/keystore> -file <full_path_to_<cert_name>.cert>
```

   If you are using the Oracle JRE bundled with PhpStorm, the default path to the truststore/keystore is `<%product_installation_folder>/jre/jre/lib/security/jssecacerts` or `<%product_installation_folder>/jre/jre/lib/security/cacerts`. Otherwise it is `<jre_home>/jre/lib/security/jssecacerts` or `<jre_home>/jre/lib/security/cacerts`.

2. When asked to enter a password for the truststore/keystore, specify the default one `changeit`.

3. Open the `PhpStorm.exe.vmoptions` file in the `<PhpStorm_installation_folder>/bin` and add the following line to it:

```
-Djavax.net.ssl.keyStore=<path to keystore>
```

4. Restart PhpStorm.

Learn more at Java6 and Java7 .

Open	Click this button to make sure that the specified server root URL address is accessible and points at the correct Web page.

Deployment: Mappings Tab

In this tab, configure mappings, that is, set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . The easiest way is to map the entire project root folder to a folder on the server, whereupon the project folder structure will be repeated on the server, provided that you have selected the Create Empty directories check box in the Options dialog box . "For more details, see Customizing Upload/Download . Below are definitions of terms used in this topic in the context of of synchronization between PhpStorm and servers.

- A local server is a server that is running in a local or a mounted folder and whose document root is NOT the parent of the project root.
- A remote server is a server on another computer (remote host).
- The server configuration root is the highest folder in the file tree on the local or remote server accessible through the server configuration. For in-place servers, it is the project root.
- A local file/folder is any file or folder under the project root.
- A remote file/folder is any file or folder on the server, either local or remote.

  Suppose you have a project `C:/Projects/My_Project/` with a folder `C:/Projects/My_Project/My_Folder` and a local server with the document root in `C:/xampp/htdocs` . You upload the entire project tree to `C:/xampp/htdocs/My_Project` . In the terms of PhpStorm, the folder `C:/Projects/My_Project/My_Folder` is referred to as local and the folder `C:/xampp/htdocs/My_Project/My_Folder` is referred to as remote.

Item	Description
Use this server as default	Click this button to have PhpStorm apply the settings of the selected configuration by default during automatic upload of changed files. This button is only enabled for the non-default servers; for the server used as default , this button is disabled.
Local Path	In this text box, specify the full path to the desired folder in the project tree. In the simplest case it is the project root. Type the path manually or click the Browse button and select the desired location in the Choose Local Path dialog box that opens.
Deployment Path	In this text box, specify the folder on the server where PhpStorm will upload the data from the folder specified in the Local Path text box. Type the path to the folder relative to the server configuration root. If the folder with the specified name does not exist yet, PhpStorm will create it, provided that you have selected the Create Empty directories check box in the Options dialog box . For more details, see Customizing Upload/Download . The text box is not available for server configurations of the type In-place.
Web Path	In this text box, type the path to the folder on the server relative to the server configuration root. Actually, type the relative path you typed in the Deployment Path text box.
Add	Click this button to have a new line added to the list of mappings.
Remove	Click this button to remove the selected mapping from the list.

Deployment: Excluded Paths Tab

Use this tab to configure a list of local and remote folders that you do not want to be involved in upload/download.

Item	Description
Add local path	Click this button to have an empty line added to the list and specify the location of the folder to be protected against upload/download. Type the path manually or click the Browse button ... and choose the required folder in the dialog that opens .
Add deployment path	Click this button to have an empty line added to the list. Click the Browse button ... The Select remote excluded path dialog box that opens shows the data on the host accessed through the selected server configuration. Select the required folder.
Remove path	Click this button to remove the selected item from the list. The button is only available when a line is selected.

Add Server Dialog

The dialog box opens when you click the Add toolbar button + on the Deployment page.

Use the dialog box to create server access configurations that give you control over interaction between PhpStorm and servers. Anytime you are going to use a server, you need to define a server access configurations, no matter whether your server is on a remote host or on your computer.

Item	Description
Name	In this text box, type the name of the new Web server configuration.
Type	From this drop-down list, choose the way to access the server. Use the Up and Down keyboard keys to scroll through the list of server configuration types. The available options are: – FTP: choose this option to have PhpStorm access the server via the FTP file transfer protocol . – SFTP: choose this option to have PhpStorm access the server via the SFTP file transfer protocol. – FTPS: choose this option to have PhpStorm access the server via the FTP file transfer protocol over SSL (the FTPS extension). – Local or mounted folder: choose this option if the Web server is running in a local or a mounted folder and its document root is NOT the parent of the project root. – In-place: choose this option if the Web server is running on your computer, your project is under its document root, and you do your development directly on the server.

When editing the server configuration name in the Name text box, use the Up and Down keys on your keyboard to change the preselected server access to type in the Type drop-down list.

Options

File | Settings | Build, Execution, Deployment | Deployment | Options for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Deployment | Options for OS X

`Ctrl+Alt+S`

Tools | Deployment | Options

Use this page to specify additional configuration settings for uploading and downloading project files to and from local and remote servers. For more details about various server access configurations, see Working with Web Servers: Copying Files .

The options specified in this dialog box apply to all defined server configurations regardless of the server type (local, remote) and the data transfer protocol used. Protocol-specific options for server configurations of the type FTP/SFTP/FTPS are defined in the Advanced Options Dialog .

Item	Description
Exclude items by name	In this text box, specify patterns for the names of files and folders that you do not need to be deployed. Use semicolons as delimiters. Wildcards are welcome. The exclusion is applied recursively. This means that if a matching folder has subfolders, the contents of these subfolders are not deployed either.
Operations logging	Use this drop-down list to specify how much detailed logging you need to have. The available options are: – Errors only: select this option to have the log show only errors occurred during upload. – Brief: select this option to have all events reflected in the log but without details. – Detailed: select this option to have more details on the upload shown in the log, for example, full file paths.
Stop operation on the first error	Select this check box to have data transfer stopped as soon as an error occurs.
Overwrite up-to-date files	Do one of the following: – Select this check box to have all the files uploaded no matter whether they have been changed since the previous upload or not. – Clear the check box to upload only files that have been changed since the previous upload.
Preserve files timestamps	Select this check box to prevent resetting timestamps of files on upload.
Delete target items when source ones do not exist	If this check box is selected, any file in the destination directory will be removed if the file with this name is not involved in the current upload. This option is applicable when synchronization if performed from the Project tool window or from the Remote Host tool window.
Create empty directories	Select this check box to have an empty directory on the server created automatically if a new local directory has been created in your project since the last upload in the source folder.
Prompt when overwriting or deleting local items	Select this check box to have PhpStorm ask you for confirmation before overwriting or deleting local items for synchronization during download.
Upload changed files automatically to the default server	From this drop-down list, choose when you want PhpStorm to automatically upload a file to the default server. The available options are: – Always: choose this option to have a file uploaded upon each save, no matter automatic or explicitly invoked. – On explicit save action: choose this option to have a file uploaded after save only if this save was invoked manually by choosing File | Save all or pressing `Ctrl+S` . – Never: choose this option to suppress automatic upload. The default server configuration is appointed on the Deployment page by selecting the desired configuration in the list and clicking the Use as Default toolbar button .
Upload external changes	Select this check box to have PhpStorm upload also the local changes that were made using a third-party tool.
Override default permissions on files	Select this check box to change the default permissions assigned to uploaded files on remote hosts. Click the Browse button ... to open the Files Default Permissions dialog box, where you can manage access to uploaded files on remote hosts by assigning permissions.
Override default permissions on folders	Select this check box to change the default permissions assigned to uploaded folders on remote hosts. Click the Browse button ... to open the Folders Default Permissions dialog box, where you can manage access to uploaded folders on remote hosts by assigning permissions.
Warn when uploading over newer file	Use this drop-down list to define the version-control policy to apply when uploading files to remote hosts. Depending on this choice, PhpStorm either checks whether any changes have been made to the corresponding files on the remote host since you downloaded them or just overwrites the remote files. – No: choose this option to have the file on the remote host overwritten with its local copy silently. All the changes made to the remote file since your last synchronization will be abandoned. – Compare timestamp and size: if you choose this option, PhpStorm performs two checks: 1. Compares the sizes of the local and remote files. 2. Compares the remote file timestamp set at the moment of the last synchronization with the current remote file timestamp. If the files differ in their size or the remote file timestamps differ, PhpStorm opens a Difference Viewer for Files , where you can explore and integrate the differences. This type of check depends on the timezone setting. If the timezone setting on your local machine is different from that on the remote host, the check may be successful even though the file versions actually differ. – Compare content: when this option is chosen, PhpStorm compares the content of the local and remote files. If any diversions are detected, PhpStorm opens a Difference Viewer for Files , where you can explore and integrate the differences.
Notify about remote changes	Select this check box to receive notifications about changes on the remote host. The check box is available only when the Compare timestamp and size: or Compare content: option is selected in the Warn when uploading over newer file drop-down list.
SFTP advanced options (IDE level)	
Add new host key to known_hosts	Choose whether PhpStorm should ask about connecting to a host not mentioned in the file `known_hosts` . The following options are available: – Always: always connect and add its record to the file `known_hosts` . – Ask: ask whether PhpStorm should connect to a host and add its record to the file `known_hosts` – Never: do not connect.
Hash hosts in	If this check box is selected, the new host record will be stored in hash format.

Advanced Options Dialog

The dialog box opens when you click the Advanced Options button in the Connection tab of the Deployment page. Use this dialog box to customize upload/download by specifying additional protocol-specific options for server configurations of the type FTP/SFTP/FTPS.

Item	Description	Available for
Passive mode	Select this check box to set the client on your machine to the passive mode , when it connects to the server to inform about being in the passive mode, receives the port number to listen to, and established data connection through the port with the received number. This mode is helpful when your machine is behind a firewall.	FTP, FTPS
Show and process hidden files	When this check box is selected: 1. Hidden files and directories are shown in the Remote Host Tool Window . 2. Hidden files and directories are involved in diff and synchronization operations. The name of a hidden file or directory starts with a dot (`.`).	FTP, FTPS
Compatibility mode	Select this check box to ensure compatibility in child file naming with your FTP server. This option is helpful if the remote FTP server reports the following error: `Invalid descendant file name <file name>` Selecting this option may slow down synchronization with the server.	FTP

Retrieve accurate files timestamps	Use this drop-down list to specify the MDTM FTP command calling policy to retrieve the last-modified time of a given file on the remote host. The available options are:
	– Always - select this option to have MDTM called for every file shown in the Remote Host tool window.
	– On copy - select this option to have MDTM called in the following cases:
	  – To check whether a file is up to date when the Overwrite up-to-date files check box in the Options dialog box is cleared.
	  – To preserve the actual time stamp of a file during download.
	– Never - select this option to suppress calling MDTM.
		FTP, FTPS, SFTP

Limit concurrent connections	Select this check box to have PhpStorm restrict the number of connections to be supported simultaneously and specify the maximum number of allowed connections in the text box.	FTP, FTPS, SFTP
Control encoding	In this text box, specify the encoding that matches the encoding used by your server. Accept the default value if you are not sure that it supports UTF-8 encoding.	FTP, FTPS, SFTP
Always use LIST command	Select this check box to use the standard `LIST` command for listing instead of the `MLSD` command. This lets you avoid problems, for example, failure during upload with the Invalid descendent file name exception if the FTP server supports `MLSD` and returns `cdir` .	FTP, FTPS
Send keep alive message each	In this text box, specify how often you want PhpStorm to send commands to the server to reset the timeout and thus preserve the connection.	FTP, FTPS, SFTP
Use keep alive command	From this drop-down list, choose the commands to be sent to the server to reset the timeout and thus preserve the connection.	FTP, FTPS
Ignore info messages	On some SFTP servers, the SSH banner may be enabled. Every time a connection is established, a pop-up window with an information message may be shown and to continue you would need to click OK. To suppress showing the information pop-up window, select the Ignore info messages check box.	SFTP

Files/Folders Default Permissions Dialog

This dialog opens when you select the Override default permissions on files or Override default permissions on folders check box in the Options dialog and click the Browse button next to it.

Use this dialog box to re-assign default server permissions to owners of files or folders, groups of owners, and other users.

The following identifiers are used for permission types:

- R stands for Read.
- W stands for Write.
- X stands for Execute

Item	Description
Owner	In this row, specify what an owner of a file or folder may do by selecting the check boxes below the corresponding identifiers.
Group	In this row, specify what a group of owners of a file or folder may do by selecting the check boxes below the corresponding identifiers.
Others	In this row, specify what any one else may do by selecting the check boxes below the corresponding identifiers.
Octal	In this text box, specify the octal representation of the specified set of permissions. By default, PhpStorm calculates and re-calculates the value of the field as you select or clear the desired check boxes. You can also specify the octal value manually.

Docker

File | Settings | Build, Execution, Deployment | Docker for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Docker for OS X

For this page to be available, the Docker integration plugin must be installed.

Docker

Specify the settings for accessing the Docker Remote API and Docker Compose.

Item	Description
Name	The name of the configuration.
API URL	The Docker Remote API URL. The default is `https://192.168.99.100:2376`.
Certificates folder	The path to the certificates folder. (The certificates are generated automatically when you start the Docker daemon for the first time.) The default is `<your_home_directory>\.docker\machine\machines\default`.
Docker Compose executable	`docker-compose` or an actual path to `docker-compose.exe` (normally located in the Docker Toolbox installation folder). The default setting `docker-compose` for Docker Compose executable is fine if: – The actual name of the executable file is `docker-compose`. – The path to the directory where the file is located is included in the environment variable `Path`. See Docker Compose.

Import credentials from Docker Machine	If you want to import credentials for accessing the Docker Remote API from Docker Machine, select the check box and specify the associated settings:
	– Docker Machine executable. `docker-machine` or an actual path to `docker-machine.exe` (normally located in the Docker Toolbox installation folder). The default setting `docker-machine` is fine if:
	  – The actual name of the executable file is `docker-machine`.
	  – The path to the directory where the file is located is included in the environment variable `Path`.
	
	  Detect. When you click Detect, PhpStorm sets Docker Machine executable to `docker-machine`, tries to detect the executable file and, if a success, shows its version.
	– Machine. After specifying the Docker Machine executable or clicking Detect, the list contains the Docker Machines available locally, and you can select the one to be used.

See also, Docker.

Toolbar

Use + to create configurations and – to delete them.

Docker Registry

File | Settings | Build, Execution, Deployment | Docker Registry for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Docker Registry for OS X

For this page to be available, the Docker integration plugin must be installed.

This page lets you manage your Docker Registry configurations which represent your Docker image repository user accounts.

Toolbar

Icon	Description
+	Create a new configuration.
−	Delete the selected configurations.

Configuration settings

Item	Description
Name	The name of the configuration
Address	The image repository service URL, e.g. − registry.hub.docker.com for Docker Hub − quay.io for Quay
Username	The user name for your user account.
Password	Your password.
Email	The email address that you specified when creating your user account.
Server	The associated Docker Remote API configuration (used to connect to the service to check that your user account settings are correct).

Coverage

File | Settings | Build, Execution, Deployment | Coverage for Windows and Linux

PhpStorm | Preferences | Build, Execution, Deployment | Coverage for OS X

Ctrl+Alt+S

Use this page to define the way the coverage data will be processed.

Item	Description
When new coverage is gathered	
Show options before applying coverage to the editor	Choose this option to show the Code Coverage dialog every time you launch a new run configuration with code coverage. The coverage options dialog is displayed, when different coverage data have been produced.
Do not apply collected coverage	Choose this option to cancel applying the new code coverage results.
Replace active suites with the new one	Choose this option to have the active suites replaced with the new one every time you launch a new run configuration with code coverage.
Add to active suites	Choose this option to have the new code coverage suite added to the active suites every time you launch a new run configuration with code coverage.
Activate Coverage View	Select this check box to have the Coverage tool window opened automatically when an application or test is run with coverage.

Languages and Frameworks

File | Settings | Languages and Frameworks for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks for OS X

When you select the Languages and Frameworks category in the left-hand pane, its main subcategories are listed in the right-hand part of the dialog.

- JavaScript
- PHP
- Schemas and DTDs
- Compass
- JSON Schema
- Node.js and NPM
- TypeScript
- SQL Dialects
- XSLT
- XSLT File Associations
- Dart

JavaScript

File | Settings | Languages and Frameworks | JavaScript for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript for OS X

`Ctrl+Alt+S`

Use the pages in this section to specify the JavaScript language version for your project and configure JavaScript support in it.

In this part:

- JavaScript. Libraries
  - JavaScript. Usage Scope
- Code Quality Tools
  - JSHint
  - JSLint
  - Closure Linter
  - JSCS
  - ESLint
- Templates
- Bower
- Yeoman
- Meteor
- PhoneGap/Cordova

Item	Description
JavaScript Language Version	From this drop-down list, choose the JavaScript engine implementation to use in your project in accordance with the targeted environment. When you select an item in the list, the icons on the right of the field indicate the browsers that suit the selected JavaScript version, that is, support the features of the selected JavaScript engine implementation. The available options are: – ECMAScript 3 – ECMAScript 5.1 – JavaScript 1.8.5 – ECMAScript 6 – ReactJS – Flow
Prefer Strict mode	Select this check box to have the strict mode standard applied to JavaScript code. This helps improve your code by enforcing best practices and suppressing insecure ones. You can also configure the type of quotes to be used in `use strict` statements by choosing Double quotes or Single quotes from the String literal type drop-down list on the Code Style. JavaScript page of the Settings dialog box.
Only type-based completion	– When this check box is cleared, the completion list contains multiple variants in complicated cases. – When the check box is selected, the completion list contains only relevant variants. By default, the check box is cleared.

JavaScript. Libraries

File | Settings | Languages and Frameworks | JavaScript | Libraries for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Libraries for OS X

`Ctrl+Alt+S`

Use this page to set up additional JavaScript libraries to expand the basic assistance provided through the JavaScript plugin . Note that the JavaScript libraries are global.

In this section:

- Libraries
- Buttons
- New Library / Edit Library Dialog Box

Libraries

Item	Description
Enabled	Each check box in this column shows whether the corresponding library is avaiable or not: – A check box is selected when the corresponding library is defined and enabled in the current project. – A check box is marked with a Dash (`-`) if the corresponding library is defined and enabled on a more detailed level, for example, in a directory or in a file. – A check box is cleared when the corresponding library is disabled.
Name	This column shows the names of the directories defined for a project.
Type	This column shows library types.

Buttons

Item	Description
Add	Click this button to create a new JavaScript library in the New Library dialog box. Refer to the section Configuring JavaScript Libraries .
Edit	Click this button to change the name and contents of the selected library in the Edit Library dialog box.
Remove	Click this button to delete the selected library.
Download	Click this button to open the Download Library dialog box, where you can have PhpStorm download and install one of the popular official JavaScript-related libraries, such as: – Dojo – ExtJS – jQuery – jQuery UI – Prototype – etc. Besides the above listed official libraries, you can download stubs for TypeScript definition files . Choose the group of libraries in the drop-down list. The available options are Official libraries and TypeScript community stubs. Depending on your choice, PhpStorm displays a list of available libraries. Select the one to be downloaded and installed, and click Download and Install. You return to the JavaScript Libraries page where the new library is added to the list. CLick OK to save the settings.
Manage scopes	Click this button to configure libraries to be used for specific files and/or directories in the JavaScript. Usage Scope dialog box.

New Library / Edit Library Dialog Box

Item	Description
Name	Specify the library name.
Framework type	From this drop-down list, choose the framework to configure as a library.
Version	In this text box, specify the version of the selected framework to use.
Visibility	In this area, specify where you want the library to be available for associating with files and folders. The available options are: – Current project: when this option is chosen, the library can be associated with files and folders within the current project only. If you later try to use the framework with another project, you will have to configure the library anew. – Global: choose this option to enable associating the library with any of your PhpStorm projects.
Files	In this section, set up the library contents.
+ (Add)	Click this button to attach a JavaScript file or directory from the file system.
– (Remove)	Click this button to detach the selected file or directory from a library.
Name	This read-only column shows the name of the selected library file or the names of relevant library files from the selected directory.
Type	Click the column to show the drop-down list of the available versions of library files or directories: debug or release.

PhpStorm enables you to create a library containing just one `.js` file, if this file is located on the Internet and can be accessed over HTTP. If you refer to a JavaScript library that is not yet available locally, but is available online, use the Download library intention action :

The library will be placed to the user home directory, and will appear in the list of configured libraries in the JavaScript - Libraries page of the Settings dialog.

JavaScript. Usage Scope

File | Settings | JavaScript | Libraries - Manage Scopes for Windows and Linux

PhpStorm | Preferences | JavaScript | Libraries - Manage Scopes for OS X

Ctrl+Alt+S

The dialog box opens when you click the Manage Scopes button in the JavaScript. Libraries page.

Use this dialog box to define the scopes where the JavaScript libraries should apply for code completion, highlighting and navigation. The scopes may cover the whole project, directories and even individual files. This helps make JavaScript code completion more precise, and avoid too long suggestion lists.

Item	Description
File/Directory	This column displays the project tree view.
Library	This column displays libraries for a file or directory, if applicable. If a library can be specified for a certain node of the project tree view, click the Library column for a selected file or directory, and choose the desired library from the list of available libraries. If JavaScript libraries are not applicable to a particular node, 'N/A' is shown in grey font.

Code Quality Tools

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools for OS X

`Ctrl+Alt+S`

Use the pages under this node to enable the built-in JavaScript code verifiers and configure their behaviour.

In this part:

- JSHint
- JSLint
- Closure Linter
- JSCS
- ESLint

JSHint

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | JSHint for Windows nd Unix

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | JSHint for OS X

Use this page to enable the built-in or downloaded JavaScript JSHint code verifier and configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have JSHint applied to verify the code in the current project. After that the other controls in the page are enabled.
Use config files	Select this check box to have the code verified according to the settings from a configuration file. A configuration file is a JSON file with the extension `.jshintrc` that specifies which JSHint options should be enabled or disabled. PhpStorm will look for a `.jshintrc` file in the working directory. If the search fails, PhpStorm will search in the parent folder, then again in the parent folder. The process is repeated until PhpStorm finds a `.jshintrc` or reaches the project root. To have PhpStorm still run verification in this case, specify the default configuration file to use. Starting with PhpStorm version 6.0.1, the globals setting in `.jshintrc` is supported. For earlier versions, a Predefined (, separated)" tree node is displayed in the bottom of the tree in the UI. When this check box is selected, the Options area is hidden and the default verification settings are unavailable.
Version	Use this drop-down list to choose the version of the tool to apply. PhpStorm comes bundled with version 1.0.0, which is used by default. PhpStorm provides the ability to download another version, which is not bundled. Actually, the alternative version is downloaded only once, whereupon it is available without download.
Options	In this area, configure JSHint behaviour by enabling or disabling JSHint options . To enable or disable an option, select or clear the corresponding check box respectively. The controls in the area fall into two groups: − Enforcing options: select the check boxes in this group to enable very strict behaviour of the verification tool and thus allow only safe JavaScript. − Relaxing options: select/clear the check boxes in this area to suppress warnings when certain types of discrepancies are detected. − Environments: select/clear these check boxes to specify for which environments you want global properties predefined. The area is available only when the Use config files

JSLint

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | JSLint for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | JSLint for OS X

`Ctrl+Alt+S`

Use this page to enable the built-in JavaScript JSLint code verifier and configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have JSLint applied to verify the code in the current project. After that the other controls in the page are enabled.
Tolerate	In this area, specify the discrepancies you want JSLint to consider worth reporting by enabling or disabling JSLint options . By default, discrepancies of all types are considered worth reporting. To have the tool skip discrepancies of a certain type, select the check box next to this type.
Assume	In this area, specify for which environments you want global properties predefined.
Stop on first error	Select this check box to have the verification process suspend as soon as the first valuable discrepancy is detected.
Safe Subset	Select this check box to have ADSafe safe subset rules enforced.
Verify ADsafe	Select this check box to have the ADSafe rules enforced.
Indentation	In this tet box, type the number of spaces to be used for indentation. This setting corresponds to the `indent` option.
Maximum line length	In this text box, limit the number of characters acceptable in one line. This setting corresponds to the `maxlen` option.
Maximum number of errors	In this text box, specify the maximum number of warnings that can be reported. The default setting is 50. This setting corresponds to the `maxerr` option.
Predefined	In this text box, specify the predefined global variables by names or through object keys. Use commas (`,` > as separators. Note This setting corresponds to the `predef` option.
Validate also	In this area, enable or disable additional verification of HTML, CSS, or JSON context and configure the checking mode, when enabled. – HTML: select this check box to have HTML context verified. Specify how you want HTML event handlers and HTML fragments treated: by default, they are reported as errors. To have the tool skip them, select the Tolerate HTML event handlers and Tolerate HTML fragments check boxes respectively. – CSS: select this check box to have CSS context verified. Specify how you want CSS workarounds treated: by default, they are reported as errors. To have the tool skip them, select the Tolerate CSS workarounds check box. – JSON: select this check box to have HTML context verified.

Closure Linter

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | Closure Linter for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | Closure Linter for OS X

`Ctrl+Alt+S`

Use this page to enable the JavaScript Closure Linter code verifier and to configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have Closure Linter applied to verify the code in the current project. After that the other controls in the page are enabled.
Closure Linter executable file	In this text box specify the path to the Closure Linter executable file: – `<Python_home>\Scripts\jslint.exe` for Windows – `/usr/local/bin/gjslint` for Linux and OS X
Configuration file	In this text box, specify the location of the previously created configuration text file to apply.

JSCS

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | JSCS for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | JSCS for OS X

`Ctrl+Alt+S`

Use this page to enable the JavaScript JSCS code verifier and to configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have JSCS applied to verify the code in the current project. After that the other controls in the page are enabled.
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
JSCS Package	In this field, specify the location of the `jscs` package installed in the current project, see Installing JSCS .
Configuration File	In this area, appoint the configuration to use. By default, PhpStorm first looks for a `jscsConfig` property in the `package.json` file of the current project. If no such property is found, PhpStorm looks for a `.jscsrc` or a `.jscs.json` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. Accordingly, you have to define the configuration to apply either as a `jscsConfig` property in the `package.json` file or in a `.jscsrc` or a `.jscs.json` configuration file, or in a custom JSON configuration file. You can also apply a predefined set of rules, either independently or in combination with a configuration file. In the latter case, the rules from the configuration file override the predefined rules. – To have PhpStorm look for a `jscsConfig` property in the `package.json` file or for a `.jscsrc` or a `.jscs.json` file, choose the Search for config(s) option. – To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the button and select the relevant file from the dialog box that opens. – To have a predefined set or rules applied, choose the desired set from the Code Style Preset drop-down list.
Code Style Preset	From this drop-down list, choose the set of predefined rules associated with the code style you use.

ESLint

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | ESLint for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | ESLint for OS X

`Ctrl+Alt+S`

Use this page to enable the JavaScript ESLint code verifier and to configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have ESLint applied to verify the code in the current project. After that the other controls in the page are enabled.
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
ESLint Package	In this field, specify the location of the `eslint` package installed in the current project, see Installing ESLint .
Configuration file	In this area, appoint the configuration to use. By default, PhpStorm first looks for a `.eslintrc` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. If no `.eslintrc` file is found, ESLint uses its default embedded configuration file. Accordingly, you have to define the configuration to apply either in a `.eslintrc` configuration file, or in a custom JSON configuration file, or rely on the default embedded configuration. – To have PhpStorm look for a `.eslintrc` file, choose the Search for .eslintrc option. If no `.eslintrc` file is found, the default embedded configuration file will be used. – To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the ... button and select the relevant file from the dialog box that opens.
Additional Rules Directory	In this field, specify the location of the files with additional code verification rules. These rules will be applied after the rules from `.eslintrc` or the above specified custom configuration file and accordingly will override them.
Extra ESLint Options	In this field, specify additional command line options to run ESLint with using spaces as separators. See ESLint Command Line Interface Options for details.

Templates

File | Settings | JavaScript - Templates for Windows and Linux

for OS X

Ctrl+Alt+S

On this page, configure processing of EJS inclusions, Handlebars expressions, and Mustache templates in PhpStorm.

Item	Description

Handlebars/Mustache	In this area, configure processing of Handlebars expressions and Mustache templates in PhpStorm. The settings specified on this page apply to dedicated Handlebars and Mustache files that have the extension `.hbs` or `.mustache` respectively.
	– Automatically insert closing tag:
	  – When this check box is selected, PhpStorm automatically inserts the second closing curly brace (`}`) of a Handlebars expression as soon as you type the first closing one.
	  PhpStorm also recognizes triple stashes (`{{{`) that prevent escaping values inside expressions. In this case, PhpStorm automatically inserts two closing curly braces as soon as you type the first closing one.
	  – When this check box is cleared, you have to type the closing curly braces and triple stashes manually.
	– Enable formatting:
	  – Select this check box to have Handlebars expressions and Mustache templates automatically reformatted during code generation, refactoring, or reformatting (Ctrl+Alt+L).
	  – Clear the check box to have the original formatting of Handlebars expressions and Mustache templates preserved.
	– Open HTML files as Handlebars/Mustache:
	  – When this check box is selected, files with the `.html` extension are treated as Handlebars/Mustache files so PhpStorm recognizes and processes Handlebars expressions and Mustache templates. The extensions of files remain unchanged but file names are supplied with the icons.
	  – When this check box is cleared only files with the `.hbs` extension are treated as Handlebars/Mustache files and Handlebars expressions and Mustache templates within them are recognized and processed.
	– Language for comments: From this drop-down list, select the language to inherit the style for comments from. When you enter a line or block comment by pressing Ctrl+Slash or Ctrl+Shift+Slash , PhpStorm inserts the comment delimiters that are used in the chosen language, for example, `{{!----}}` for Handlebars, `/**/` for JavaScript, `<!---->` for HTML, etc.
	
	For more details about Handlebars expressions and Mustache templates see http://handlebarsjs.com/ .
	The area is only available when the Handlebars/Mustache plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

EJS	In this area, specify delimiters for inclusions of EJS in your code. Based on these delimiters, PhpStorm will recognize and process such inclusions correctly. A pair of delimiters is also referred to as execute tag. – EJS open delimiter: In this text box, type the character string that will indicate the beginning of an EJS inclusion. The default delimiter is `<%` . – EJS close delimiter: In this text box, type the character string that will indicate the end of an EJS inclusion. The default delimiter is `%>` .

Bower

File | Settings | Languages and Frameworks | JavaScript | Bower for Widnows and Unix

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Bower for OS X

On this page, specify the location of the Bower package manager executable and configuration files and install, uninstall, or upgrade external tools through it.

Item	Description
Node interpreter	Specify here the location of the Node.js executable. Type the path manually, or choose it from the drop-down list if it was previously used, or click the Browse button ... and choose the location in the dialog box that opens. See Node.js for details.
Bower executable	In this field, specify the location of the Bower executable file (`bower.cmd` or other depending on the operating system used). Type the path manually, or choose it from the drop-down list if it was previously used, or click the Browse button ... and choose the location in the dialog box that opens.
bower.json	In this text bos, specify the location of the bower.json file. Type the path manually, or click the Browse button ... and choose the location in the dialog box that opens.
Dependencies	The area shows a list of all the -dependent packages that are currently installed on your computer. – Package: this read-only field shows the name of a package, exactly as it should be referenced if you were installing it in the command line mode. – Version: this read-only field shows the version of the package installed on your computer. – Latest: this read-only field shows the latest released version of the package. If a package is not up-to-date, it is marked with a blue arrow. – Click + to have a new package installed. In the Available Packages dialog box that opens, select the relevant package. Click Install Package when ready. – Click — to have the selected package removed. – Click to have the current version of the selected package replaced with the latest released version. The button is enabled only when the selected project is not up-to-date.

Yeoman

File | Settings | Languages and Frameworks | JavaScript | Yeoman for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Yeoman for OS X

`Ctrl+Alt+S`

The page is available only when the Yeoman and the NodeJS plugins are installed and enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

On this page, specify the location of the Node.js executable file and the Yeoman project generator package installed through Node Package Manager.

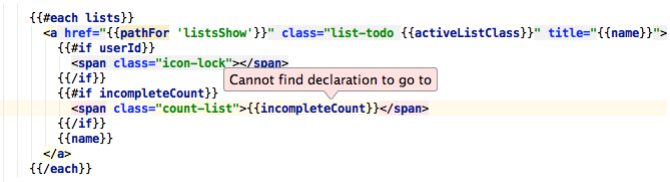
Item	Description
Node interpreter	Specify here the location of the Node.js executable. Type the path manually, or choose it from the drop-down list if it was previously used, or click the Browse button ... and choose the location in the dialog box that opens. See Node.js for details.
Yeoman yo package	In this field, specify the location of the global `yo` package, see Installing and Removing External Software Using Node Package Manager , section Installing an External Tool Globally.

Meteor

File | Settings | Languages and Frameworks | JavaScript | Meteor for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Meteor for OS X

The page is available when the Meteor plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Item	Description
Meteor executable	In this field, specify the location of the Meteor executable file (see Installing Meteor).
Automatically exclude ".meteor/local" directory on open project	– By default, the check box is selected and the `.meteor/local` folder, which is intended for storing the built application, is automatically marked as excluded and is not involved in indexing. – Clear the check box to have PhpStorm show the `.meteor/local` folder and its contents in the Project tool window. By default, excluded files are shown in the project tree. To have the `.meteor/local` folder hidden, click the ⚙ button on the toolbar of the Project tool window and remove a tick next to the Show Excluded Files option.
Enable Meteor 'Hot code push'	– Select this check box to activate the native Meteor hot code pushes functionality and thus have the changes made to have the client-side code during a debugging session applied. By default, the check box is selected. Changes made to the server-side code are uploaded through the PhpStorm Live Edit functionality, see Configuring the Update Policy for the Server Side Code . – When the check box is cleared, PhpStorm does not provide any way to apply the changes made to the client-side code during a debugging session.
Automatically import Meteor packages as external library	– When the check box is selected, PhpStorm automatically imports the external packages from the `meteor/packages` file. As a result, PhpStorm provides full range coding assistance: resolves references to Meteor built-in functions, for example, `check(true)` , and to functions from third-party packages, provides proper syntax and error highlighting, supports debugging with source maps, etc. – When this check box is cleared, PhpStorm does not automatically import the external packages from the `meteor/packages` file. As a result no coding assistance is provided. To improve the situation, open the `meteor/packages` file in the editor and click the Import packages as library link or run the `meteor --update` command. By default, the check box is selected.
Weak search for Spacebars templates	– Select this check box to enable PhpStorm to search inside Spacebars templates. When this check box is selected and you invoke the Go to Declaration action on a helper, PhpStorm displays a list of all occurrences of this helper in templates. Choose the relevant one from the list. – When you invoke Got to Declaration with this check box cleared, PhpStorm does not perform any search and displays a tooltip which the following message: Cannot find declaration to go to. 

PhoneGap/Cordova

File | Settings | Languages and Frameworks | JavaScript | PhoneGap/Cordova for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | PhoneGap/Cordova for OS X

`Ctrl+Alt+S`

The page is available when the PhoneGap/Cordova plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Item	Description
PhoneGap/Cordova Executable Path	In this field, specify the location of the executable file `phonegap.cmd` , `cordova.cmd` , or `ionic.cml` (see Installing PhoneGap/Cordova/Ionic).
PhoneGap/Cordova Version	This read-only field shows the installed PhoneGap/Cordova/Ionic version that PhpStorm detects automatically.
PhoneGap/Cordova Working Directory	In this field, specify the folder under which the PhoneGap/Cordova/Ionic application files to run are stored.
Automatically exclude 'platforms' directory on open project	Select this check box to have the platforms directory marked as excluded automatically when you open the project. As a result, PhpStorm ignores it during indexing, parsing, and code completion.
Plugins	In this area, configure a list of plugins to use in your development by installing required packages. The list shows all the PhoneGap/Cordova/Ionic plugins that are currently installed on your computer, both at the global and at the project level. – To install a plugin, click the Install button +. In the Available Packages dialog box that opens, select the required package. To have the plugin installed globally so it is accessible from all your PhpStorm projects, select the Options check box and type `--global` in the text box. Click Install Package. – To remove a plugin, select it in the list and click the Uninstall button —. – To upgrade a plugin to the latest available version, select the plugin in the list and click the Upgrade button ↑. See Apache Cordova Plugins and PhoneGap Plugins for information about plugins and their use.

PHP

File | Settings | Languages and Frameworks | PHP for Windows and Linux

for OS X

Ctrl+Alt+S

The page and all the pages under this node are available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to configure PHP development and unit testing support in the project by choosing one of the available PHP interpreters, see Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters .

ItemTooltip and Shortcut	Description
PHP language level	In this drop-down list, specify the PHP functionality scope to get coding assistance for. Each functionality scope is associated with PHP version that supports this functionality. Currently PHP 5.3, PHP 5.4, PHP 5.5, PHP 5.6, PHP 7, and PHP 7.1 levels are supported. No correlation between the PHP version used in the project and the language level is enforced. Although the language version of each interpreter is detected automatically, you can still tell PhpStorm to provide you with coding assistance that corresponds to another language level. However, if you attempt to use a code construct that is not supported by the specified language level, PhpStorm suggests a Switch to PHP <version> quick-fix .
Interpreter	From this drop-down list, choose the PHP interpreter to use in the current project. The list contains all the currently configured local and remote PHP interpreters. See Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters for details.
Reload	Click this button to make sure that the configuration you have chosen points at the relevant installation. If no PHP executable is detected at the specified directory, PhpStorm displays the corresponding error message.
Show phpinfo	Click this button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed.
Shift+Enter	Click this button next to the Interpreter drop-down list to create a new PhpStorm-wide PHP installation configuration in the Interpreters dialog box, that opens, see Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters .
Path mappings	
Include path	The area shows a list of configured include paths. Include paths are used for code completion and reference resolution in some functions/methods that use file paths as arguments (for example, `require()` or `include()`). You also need to configure PEAR as an include path to run the PHPUnit test framework from it when the Load from include path option is chosen on the PHPUnit page of the Settings / Preferences Dialog . Use the Add + and Remove — buttons to manage the contents of the list. Use the Up and Down buttons to change the order of items in the list.

Interpreters

File | Settings | Languages and Frameworks | PHP for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Browse button next to the Interpreter drop-down list in the Development environment section of the PHP page.

Use this dialog box to configure PHP engines as interpreters, see Configuring Local PHP Interpreters and Configuring Remote PHP Interpreters . In this dialog box, you can add new interpreters and edit or remove the existing ones.

The dialog box consists of two panes. The left-hand pane shows a list of all configured PHP interpreters, both local and remote ones, and a toolbar for adding, removing, and copying PHP interpreter configurations. The contents of the right-hand pane depend on the type of the interpreter selected in the left-hand pane. Use the control in this pane to edit the settings of the selected interpreter and even choose its type, if necessary.

- Left-hand pane
- Right-hand pane
- Configuration directives dialog

Left-hand pane

The left-hand pane shows a list of all configured PHP interpreters, both local and remote ones, and a toolbar for adding, removing, and copying PHP interpreter configurations.

ItemTooltipDescription		
+	Add	Click this button to add a new PHP interpreter to the list. From the drop-down list, choose the type of the interpreter (Local or Remote), and configure a local interpreter in the right-hand pane or a remote interpreter in the Configure PHP Remote Interpreter Dialog dialog that opens.
−	Delete	Click this button to remove the selected interpreter from the list.
	Copy	Click this button to create a new interpreter with the settings copied from the selected one.

Right-hand pane

The contents of the right-hand pane depend on the type of the interpreter selected in the left-hand pane. Use the control in this pane to edit the settings of the selected interpreter and even choose its type, if necessary.

Type of the selected interpreter	Description
Local	– Name: In this text box, type the identifier to distinguish the interpreter from others, for example, `php_installation_<version>` . – PHP executable: In this text box, specify the path to the PHP engine. Type the path manually or click the Browse button and choose the executable file in the dialog box that opens. PhpStorm detects the version of PHP and displays it in the PHP version read-only field. PhpStorm also detects the debugging engine and the `php.ini` configuration file. The type of the debugging engine associated with the PHP interpreter and its version are displayed in the Debugger read-only field. If no debugger is detected or you have disabled it in `php.ini` file (see Configuring Xdebug for Using in the On-Demand Mode), the field shows Debugger: Not installed. The location of `php.ini` is displayed in the Configuration file read-only field. To edit the `php.ini` in PhpStorm, click Open in Editor. – (Reload): Click this button to check that the specified PHP home directory actually contains a PHP executable file. If no PHP executable is detected at the specified location, PhpStorm displays the corresponding error message. – (Show phpinfo): Click this button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed. – Debugger extension: in this text box, specify the location of the Xdebug extension to enable PhpStorm to activate it when necessary. Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable debugger for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster. – Configuration Options: Use this text box to customize the configuration settings of the installation by composing a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. Click the Browse button to open the Configuration Directives dialog box and create a list of new `php.ini` entries there.

Remote interpreter accessible through SSH	– Host: in this field, type the name of the host on which the interpreter is installed.
	– Port: in this field, type the port which the SSH server on the remote host listens to. The default port number is 22.
	– User name: in the field, type the user name under which you are registered on the SSH server.
	– Auth type: from this drop-down list, choose the authentication method.
	  – To access the host through a password, choose Password from the Auth type drop-down list and type the password.
	  – To access the host through a pair of SSH keys, choose Key pair, then specify the path to the file where your private key is stored or the passphrase if you have configured it during the generation of the key pair.
	– Name: In this text box, type the identifier to distinguish the interpreter from others, for example, `php_installation_<version>` .
	– PHP executable: In this text box, specify the path to the PHP engine. Type the path manually or click the Browse button and choose the executable file in the dialog box that opens. PhpStorm detects the version of PHP and displays it in the PHP version read-only field. PhpStorm also detects the debugging engine and the `php.ini` configuration file. The type of the debugging engine associated with the PHP interpreter and its version are displayed in the Debugger read-only field. If no debugger is detected or you have disabled it in `php.ini` file (see Configuring Xdebug for Using in the On-Demand Mode), the field shows Debugger: Not installed.
	
	The location of `php.ini` is displayed in the Configuration file read-only field. To edit the `php.ini` in PhpStorm, click Open in Editor.
	– (Reload): Click this button to check that the specified PHP home directory actually contains a PHP executable file. If no PHP executable is detected at the specified location, PhpStorm displays the corresponding error message.
	– (Show phpinfo): Click this button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed.
	– Debugger extension: in this text box, specify the location of the Xdebug extension to enable PhpStorm to activate it when necessary.

	Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable debugger for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster. – Configuration Options: Use this text box to customize the configuration settings of the installation by composing a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. Click the Browse button ... to open the Configuration Directives dialog box and create a list of new `php.ini` entries there.
Remote interpreter on a Vagrant instance	– Name: In this text box, type the identifier to distinguish the interpreter from others, for example, `php_installation_<version>`. – PHP executable: In this text box, specify the path to the PHP engine. Type the path manually or click the Browse button ... and choose the executable file in the dialog box that opens. PhpStorm detects the version of PHP and displays it in the PHP version read-only field. PhpStorm also detects the debugging engine and the `php.ini` configuration file. The type of the debugging engine associated with the PHP interpreter and its version are displayed in the Debugger read-only field. If no debugger is detected or you have disabled it in `php.ini` file (see Configuring Xdebug for Using in the On-Demand Mode), the field shows Debugger: Not installed. The location of `php.ini` is displayed in the Configuration file read-only field. To edit the `php.ini` in PhpStorm, click Open in Editor. – (Reload): Click this button to check that the specified PHP home directory actually contains a PHP executable file. If no PHP executable is detected at the specified location, PhpStorm displays the corresponding error message. – (Show phpinfo): Click this button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed. – Debugger extension: in this text box, specify the location of the Xdebug extension to enable PhpStorm to activate it when necessary. Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable debugger for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster. – Configuration Options: Use this text box to customize the configuration settings of the installation by composing a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. Click the Browse button ... to open the Configuration Directives dialog box and create a list of new `php.ini` entries there.
Remote interpreter accessible through a deployment configuration	– Deployment Configuration: from this drop-down list, choose the server access configuration of the type SFTP according to which you want PhpStorm to connect to the target host. If the settings specified in the chosen configuration ensure successful connection, PhpStorm displays the URL address of the target host as a link in the Deployment Host URL field. – Name: In this text box, type the identifier to distinguish the interpreter from others, for example, `php_installation_<version>`. – PHP executable: In this text box, specify the path to the PHP engine. Type the path manually or click the Browse button ... and choose the executable file in the dialog box that opens. PhpStorm detects the version of PHP and displays it in the PHP version read-only field. PhpStorm also detects the debugging engine and the `php.ini` configuration file. The type of the debugging engine associated with the PHP interpreter and its version are displayed in the Debugger read-only field. If no debugger is detected or you have disabled it in `php.ini` file (see Configuring Xdebug for Using in the On-Demand Mode), the field shows Debugger: Not installed. The location of `php.ini` is displayed in the Configuration file read-only field. To edit the `php.ini` in PhpStorm, click Open in Editor. – (Reload): Click this button to check that the specified PHP home directory actually contains a PHP executable file. If no PHP executable is detected at the specified location, PhpStorm displays the corresponding error message. – (Show phpinfo): Click this button to have PhpStorm display a separate information window where you can examine the installation details and view the list of loaded extension and configured options. Please note that the options specified in the Configuration Options field of the Interpreters dialog box are not listed. – Debugger extension: in this text box, specify the location of the Xdebug extension to enable PhpStorm to activate it when necessary. Starting with version 2016.2, PhpStorm supports the On-Demand mode where you can disable debugger for your global PHP installation, and have it enabled automatically on demand only when you are debugging your command-line scripts or when you need code coverage reports. This lets your command line scripts (including Composer and unit tests) run much faster. – Configuration Options: Use this text box to customize the configuration settings of the installation by composing a string of configuration directives to be passed through the -d command line option and thus add new entries to the `php.ini` file. Click the Browse button ... to open the Configuration Directives dialog box and create a list of new `php.ini` entries there.

Configuration directives dialog

Item	Tooltip and Shortcut	Description
Name		In this text box, type the name of the new entry.
Value		In this text box, type the value of the new entry.
+	Add Alt+Insert	Click this button to have a new line added to the list and specify the name and value of a new entry there.
–	Remove Alt+Delete	Click this button to remove the selected entry from the list.
	Up / Down Alt+Up / Alt+Down	Use these buttons to move the selected entry up or down lin the list. The order of entries in the list determine the order in which they are passed through the -d command line option .

Configure PHP Remote Interpreter Dialog

File | Settings | Languages and Frameworks | PHP for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP for OS X

`Ctrl+Alt+S`

The dialog box is available only when the PHP Remote Interpreter plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

The dialog box opens when you click the Add toolbar button + in the left-hand pane of the Interpreters dialog box and choose Remote... from the drop-down menu.

Use this dialog box to configure access to PHP engines installed on remote hosts or in development environments set up in Vagrant instances.

Item	Description

SSH Credentials	Choose this option to configure access to a PHP interpreter on a remote host through SSH credentials. In the fields of the dialog box, specify the following:
	– Host: in this field, type the name of the host on which the interpreter is installed.
	– Port: in this field, type the port which the SSH server on the remote host listens to. The default port number is 22.
	– User name: in the field, type the user name under which you are registered on the SSH server.
	– Auth type: from this drop-down list, choose the authentication method.
	  – To access the host through a password, choose Password from the Auth type drop-down list and type the password.
	  – To access the host through a pair of SSH keys, choose Key pair, then specify the path to the file where your private key is stored or the passphrase if you have configured it during the generation of the key pair.
	
	To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm first attempts to retrieve path mappings itself by processing all the available application-level configurations. If PhpStorm finds the configurations with the same host as the one specified above, in the Host field, the mappings from these configurations are merged automatically. If no configurations with this host are found, PhpStorm displays an error message informing you that path mappings are note configured.
	
	To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the server access configuration in question, switch to the Mappings tab, and map local folders to folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.

Vagrant	This option is available only when the Vagrant repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Choose this option to configure access to a PHP interpreter installed in a Vagrant instance using your Vagrant credentials. Technically, it is the folder where the VagrantFile configuration file for the desired environment is located. Based on this setting, PhpStorm detects the Vagrant host and shows it as a link in the Vagrant Host URL read-only field. To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . PhpStorm evaluates path mappings from the `VagrantFile` configuration file.
Deployment Configuration	This option is available only when the Remote Hosts Access plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins . Choose this option to configure access to a PHP interpreter on a remote host using a server access configuration. This option is available only if you have at least one server access configuration of the type SFTP, see Creating a Remote Server Configuration . From the Deployment Configuration drop-down list, choose the server access configuration of the type SFTP according to which you want PhpStorm to connect to the target host. If the settings specified in the chosen configuration ensure successful connection, PhpStorm displays the URL address of the target host as a link in the Deployment Host URL field. To use an interpreter configuration, you need path mappings that set correspondence between the project folders, the folders on the server to copy project files to, and the URL addresses to access the copied data on the server . By default, PhpStorm retrieves path mappings from the chosen server access (deployment) configuration. If the configuration does not contain path mappings, PhpStorm displays the corresponding error message. To fix the problem, open the Deployment page under the Build, Execution, Deployment node, select the relevant server access configuration, switch to the Mappings tab, and map the local folders to the folders on the server as described in Creating a Remote Server Configuration , section Mapping Local Folders to Folders on the Server and the URL Addresses to Access Them.
PHP Interpreter Path	In this field, specify the location of the PHP executable file in accordance with the development environment configured on the selected remote host. By default PhpStorm suggests the `/usr/bin/php` folder. To specify another folder, click the Browse button ... and choose the relevant folder in the dialog box that opens. Note that the PHP home directory must be open for edit. When you click OK, PhpStorm checks whether the PHP executable is actually stored in the specified folder. – If no PHP executable is found, PhpStorm displays an error message asking you whether to continue searching or save the interpreter configuration anyway. – If the PHP executable is found, you return to the Interpreters where the installation folder and the detected version of the PHP interpreter are displayed.

Debug

File | Settings | Languages and Frameworks | PHP | Debug for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Debug for OS X

`Ctrl+Alt+S`

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to configure the behaviour of the Xdebug and Zend Debugger debugging tools.

Item	Description

Evaluation	– Show array and object children in Debug Console: Select this check box to show the output for arrays and objects in the Console pane, see Debug Tool Window. Console . When the check box is cleared, the output is not displayed.
	– Safe evaluation mode in value hints and Watches Frame:
	  – When this check box is selected, PhpStorm checks that the expression or code fragment to be evaluated does not contain any undefined elements and informs you about any discrepancies detected.
	  – If the check box is cleared, an exception appears if PhpStorm encounters any undefined elements during evaluation.
	
	See Evaluating Expressions for details.

Local debug	In this area, customize local debugging, when the processed files are on your machine. – Pass required configuration options through command line (still need to enable debug extension manually): select this check box to have debugger configuration options passed through a command line.

External Connections	In this area, specify how you want PhpStorm to treat connections received from hosts and through ports that are not registered as server configurations .
	– Ignore external connections through unregistered server configurations: Select this check box to have PhpStorm ignore connections received from hosts and through ports that are not registered as server configurations . When this check box is selected, PhpStorm does not attempt to create a server configuration automatically.
	– Detect path mappings from deployment configurations:
	  – When this check box is selected, PhpStorm attempts to retrieve path mappings for debugging in a remote environment from the server access configuration (deployment configuration).
	  – When the check box is cleared, you have to specify the path mappings manually.
	
	See Validating the Configuration of a Debugging Engine , Web Server Debug Validation Dialog , and Configuring Remote PHP Interpreters for details.
	– Break at first line in PHP scripts: Select this check box to have the debugger stop as soon as connection between it and PhpStorm is established (instead of running automatically until the first breakpoint is reached). Alternatively turn on the Run | Break at first line in PHP scripts option on the main menu.
	– Max. simultaneous connections: Use this spin box to limit the number of external connections that can be processed simultaneously.

Xdebug	Use the controls in this area to configure debugging using the Xdebug tool. – Debug port: in this text box, specify the port for PhpStorm and the Xdebug engine to communicate through. This must be exactly the same port number as specified in the `php.ini` file: `xdebug.remote_port = <port_number>` By default, Xdebug listens on port 9000. – Can accept external connections: select this check box to enable PhpStorm to accept any incoming connections from Xdebug engines through the port specified in the Debug port text box. – Force break at the first line when no path mapping is specified: Select this check box to have the debugger stop as soon as it reaches and opens a file that is not mapped to any file in the project on the Servers page. The debugger stops at the first line of this file and Debug Tool Window. Variables shows the following error message: Cannot find a local copy of the file on server <path to the file on the server> and a link Click to set up mappings. Click the link to open the Resolve Path Mappings Problem dialog box and map the problem file to its local copy. When this check box cleared, the debugger does not stop upon reaching and opening an unmapped file, the file is just processed, and no error messages are displayed. – Force break at the first line when the script is outside the project: Select this check box to have the debugger stop at the first line as soon as it reaches and opens a file outside the current project. With this check box cleared, the debugger continues upon opening a file outside the current project.
Zend Debugger	Use the controls in this area to configure debugging using the Zend Debugger tool. – Debug port: In this text box, specify the port for PhpStorm and the Zend Debugger engine to communicate through. Type the port number within the tunnel specified in the `php.ini` file through `zend_debugger.tunnel_min_port` and `zend_debugger.tunnel_max_port` . For details, see http://files.zend.com/help/previous-version/Zend-Server-4-Community-Edition/zenddebugger.html – Can accept external connections: Select this check box to enable PhpStorm to accept any incoming connections from Zend Debugger engines through the port specified in the Debug port text box. – Settings broadcasting port: In this text box, specify the port through which the debugger settings are passed to the debugging toolbar in the browser. – Ignore Z-Ray system requests: Select this check box to block requests from the Z-Ray system if they annoy you by invoking the PhpStorm debugger too often. Tip If starting the Zend Debugger tool fails with the message "Port is busy", specify a port number of your choice higher than 10000.
Use debugger bookmarklets to initiate debugger from your favorite browser	Follow this link to open the Zend Debugger & Xdebug bookmarklets page where you can generate bookmarklets through which you will start/stop a debugging session by controlling the debugger cookie .
Advanced Settings	Notify if debug session was finished without being stopped: Select this check box to have PhpStorm display a notification when no breakpoints are hit during Zero-Configuration debugging. This may happen if the path mappings are not configured or configured erroneously, or if you have not set any breakpoints. In the latter case, enabling selecting the Break at First Line in PHP Scripts check box in the External Connections area or turning the Run | Break at First Line in PHP Scripts option on the main menu may also help. If the check box is cleared, no debugging sessions is established and the PHP script is just executed without being suspended.

Skipped Paths

File | Settings | Languages and Frameworks | PHP | Debug | Skipped Paths for Windows and Linux

for OS X

Ctrl+Alt+S

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, specify the scripts requests to which you want PhpStorm to ignore during debugging. This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the skipped paths list. You can also group such scripts into folders and add these folders to the "ignore list".

Item	Description
Notify about skipped paths	Select this check box to have PhpStorm inform you every time it receives a request to a script to be skipped.
Skipped paths	This list box shows the scripts and folders to ignore requests to.
+ Add (Alt+Insert)	Click this button to have a new line added to the list. Then click the Browse button ... and in the dialog box that opens choose the file or folder to skip connections to.
– Remove (Alt+Delete)	Click this button to have the selected item removed from the list.

DBGp Proxy

File | Settings | Languages and Frameworks | PHP | Debug | DBGp Proxy for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Debug | DBGp Proxy for OS X

`Ctrl+Alt+S`

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, enable, disable, and re-configure your access to debugging PHP applications in the multiuser mode via an Xdebug proxy server .

Item	Description
IDE key	In this text box, specify the name for the Proxy server to identify connections from your IDE. This should be the value of the `xdebug.idekey` setting in your currently active `php.ini` configuration file.
Host	In this text box, specify the host on which the Xdebug proxy server resides.
Port	In this text box, specify the port which PhpStorm will listen to during a proxy debugging session.

Step Filters

File | Settings | PHP | Debug | Step Filters for Windows and Linux

PhpStorm | Preferences | PHP | Debug | Step Filters for OS X

`Ctrl+Alt+S`

The page and all the pages under this node are available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, specify the PHP methods and functions which you want the debugger to skip in addition to the common list of items to be skipped which is defined on the Stepping page.

Item	Description
Skip magic methods	Select this check box to suppress stepping into PHP magic methods and any other methods or functions with names starting with a double underscore `__` .
Skip constructors	Select this check box to suppress stepping into constructors .
Skipped Methods	In this area, create a list of specific methods that you do not want PhpStorm to step into. – To add a method to the list, click the Add button and in the Add Method dialog box that opens type the name of the required function or the fully qualified name of the required method in the format `\namespace\class->method` . – To remove a method from the list so PhpStorm steps into it during a debugging sessions, select the desired method in the list and click the Remove button.

Servers

File | Settings | Languages and Frameworks | PHP | Servers for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Servers for OS X

Ctrl+Alt+S

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, configure HTTP access for debugging engines to interact with local and remote Web servers and set correspondence between files on the server and their local copies in the PhpStorm project. The settings from debug server configurations are used when debugging with run/debug configurations of the type PHP Web Application or PHP Remote Debug and during Zero-Configuration Debugging sessions.

Toolbar and common options

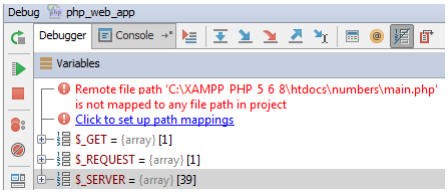
Use the toolbar buttons to manage the list of configurations.

Item	Tooltip and shortcut	Description
+	Add Insert	Click this button to define a new configuration.
−	Delete Delete	Click this button to remove the selected configuration from the list.
	Import	Click this button to open the Import from Deployment Configuration Dialog dialog box. In this dialog box, choose a configuration to access the application on the server and use the host and port settings from it. The dialog box also shows the path mappings retrieved from the deployment configuration. You need to transform relative paths on the server into absolute paths: – For an FTP, SFTP, or FTPS server access configuration, specify the absolute path to the server deployment root. This path will be added as a prefix to the path from the Root Path text box on the Deployment: Connection Tab . If you are not sure about this absolute path, you can open the Remote Host tool window, choose the required deployment configuration, position the cursor at the root folder, and choose Copy Path on the context menu, see Accessing Files on Web Servers for details. Alternatively, contact your hosting provider. – For a server access configuration of the type Local or Mounted Folder, specify the absolute path to the document root of the server or to the mounted folder. This path should be the one specified in the Folder field on the Deployment: Connection Tab . – For Inplace Server configurations no mappings are required because the local and remote paths are the same in this case. See Configuring Synchronization with a Web Server for details.

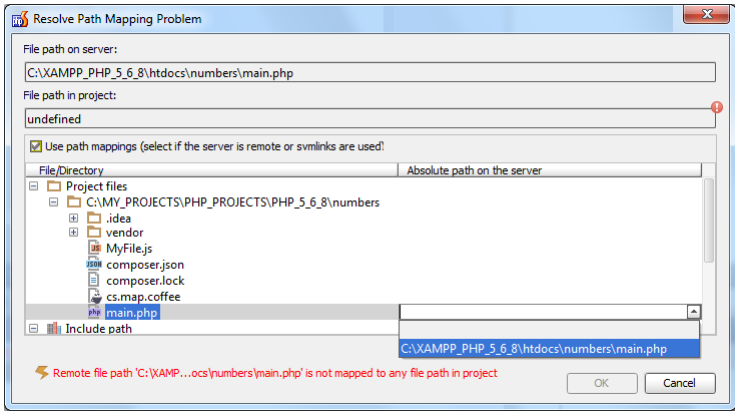
Configuration Details

In this area, specify the connection parameters and mappings to be used during debugging sessions. Note that the Validate Remote Environment button has been removed from this page, this functionality is now available through Run | Web Server Debug Validation on the main menu.

Item	Description
Name	In this text box, type the name of the server debug configuration.
Host	In this text box, type the name of the host where the target application is deployed.
Port	In this text box, type the port to connect to the specified host through. If you are using localhost on your machine, this setting should correspond with the port specified in the configuration file of the local Web server where the application will be executed or debugged.
Debugger	From this drop-down list, select the debug engine to use. The available options are: – Xdebug – Zend Debugger
Use path mappings	– Select this check box, if you are working on a remote Web server, that is, when the Web server is on a physically remote host, or the Web server is installed on your machine but your project is outside the Web server document root. Also select the check box if you are using symlinks. Map the absolute paths to the files and folders on the server with absolute paths to your project files in the local file system using the Path on server and File/Directory fields respectively. – File / Directory: This read-only field displays the files and folders of the current project. Select a file or a folder to be used as the local copy. – Path on server: In this filed, specify the absolute path to the file or folder on the target server to which the selected local file or folder corresponds. Type the path manually or select it from the drop-down list. – Clear this check box if you are working right on your Web server so your project root is under the server document root. In this case the absolute paths to the files on the Web server and the absolute paths to the corresponding files in your project are the same. If you do not specify any path mappings and start debugging an application that is not under the server document root, PhpStorm displays an error message:



The Click to set up path mappings link brings up the Resolve Path Mappings Problem dialog box, where you can define the path mappings:



When you click OK and leave the dialog box, PhpStorm selects the Use path mappings check box on the Servers page automatically.

Shared

Select this check box to share the debug server configuration across a team. The host/port settings and the path mappings are stored in the `.idea/php.xml` file is available to all team members through a version control system. Note that mappings are shared only for directories inside the project.

Import from Deployment Configuration Dialog

File | Settings | Languages and Frameworks | PHP | Servers for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Servers for OS X

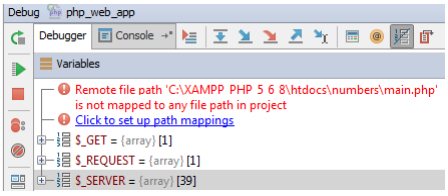
`Ctrl+Alt+S`

The dialog box is available only when the PHP and Remote Hosts Access plugins are enabled. The plugins are not bundled with PhpStorm, but they can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins . Once enabled, the plugins are available at the IDE level, that is, you can use it in all your PhpStorm projects.

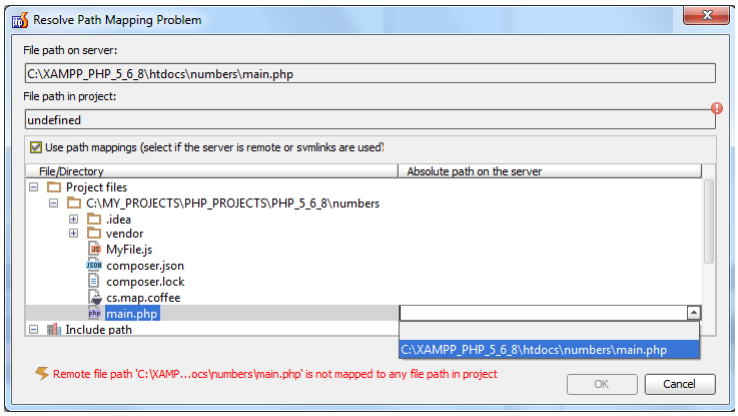
The dialog box opens when you click the Import button on the toolbar of the Servers page.

In this dialog box, choose a configuration to access the application on the server and use the host and port settings from it.

Item	Description
Deployment	From this drop-down list, choose the server access configuration (deployment configuration) to copy the server access settings from. The list contains all the available deployment configurations. To create a new configuration, click ... and specify new settings in the Deployment: Connection Tab dialog box that opens. In the text box below, specify the absolute path to the server deployment root folder, the name of the text box depends on the type of the selected server access configuration: Absolute path to the deployment root for FTP/ SFTP/ FTPS or Remote path to the mounted folder for Local or mounted folder. This path will be added as a prefix to the path from the Root Path text box on the Deployment: Connection Tab . If you are not sure about this absolute path, you can open the Remote Host tool window, choose the required deployment configuration, position the cursor at the root folder, and choose Copy Path on the context menu, see Accessing Files on Web Servers for details. Alternatively, contact your hosting provider.
Absolute path to the deployment root/ Remote path to the mounted folder	Depending on the type of the server access configuration chosen from the Deployment drop-down list, specify one of the following: – For an FTP, SFTP, or FTPS server access configuration, specify the absolute path to the server deployment root. This path will be added as a prefix to the path from the Root Path text box on the Deployment: Connection Tab . If you are not sure about this absolute path, you can open the Remote Host tool window, choose the required deployment configuration, position the cursor at the root folder, and choose Copy Path on the context menu, see Accessing Files on Web Servers for details. Alternatively, contact your hosting provider. – For Local or mounted folder, type the absolute path to the server root as specified in the Folder field of the Import from Deployment Configuration Dialog dialog box. – For Inplace Server configurations no mappings are required because the local and remote paths are the same in this case.
Preview	The area shows the host/port and the path mappings retrieved from the chosen server access configuration (deployment configuration). When you choose the deployment configuration to use, the Absolute path on the server field shows relative paths mapped to the project files and folders in the chosen configuration, that is, paths to files and folders relative to the deployment root. As you specify the absolute path to the deployment root (the server root for FTP/SFTP/FTPS or the mounted folder), the contents of the field are updated automatically and finally the field shows absolute paths on the server. Specify how PhpStorm will set up correspondence between files on the server and their local copies. Based on these mappings, PhpStorm will open local copies of currently processed files. Path mappings in PHP Debug Server configurations look very similar to the path mappings in server access (deployment configurations. Unfortunately, they cannot be reused, as deployment configurations uses relative paths while PHP Debug Servers configurations rely on absolute paths. – Select the Use path mappings check box if you are working on a remote Web server, that is, when the Web server is on a physically remote host, or the Web server is installed on your machine but your project is outside the Web server document root. Also select the check box if you are using symlinks. Map the absolute paths to the files and folders on the server with absolute paths to your project files in the local file system using the Path on server and File/Directory fields respectively. – File / Directory: This read-only field displays the files and folders of the current project. Select a file or a folder to be used as the local copy. – Path on server: In this filed, specify the absolute path to the file or folder on the target server to which the selected local file or folder corresponds. Type the path manually or select it from the drop-down list. – Clear the Use path mappings check box if you are working right on your Web server so your project root is under the server document root. In this case the absolute paths to the files on the Web server and the absolute paths to the corresponding files in your project are the same. If you do not specify any path mappings and start debugging an application that is not under the server document root, PhpStorm displays an error message:



The Click to set up path mappings link brings up the Resolve Path Mappings Problem dialog box, where you can define the path mappings:



When you click OK and leave the dialog box, PhpStorm selects the Use path mappings check box on the Servers page automatically.

Behat

File | Settings | Languages and Frameworks | PHP | Behat for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Behat for OS X

`Ctrl+Alt+S`

This page is available only when the PHP and Behat plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Before you start, make sure the PHP interpreter is configured in PhpStorm on the PHP page , as described in Enabling PHP Support .

On this page, configure the Behat framework to enable behaviour-driven development in PhpStorm.

Item	Description
Interpreter	In this field, PhpStorm displays the appointed PHP interpreter configuration to use with Behat. The field is read-only, to update the chosen interpreter configuration or create a new one, click the ... whereupon PhpStorm brings you to the Interpreters dialog box .
Behat Library	In this area, specify the Behat installation to use. – Path to Behat directory of phar file: in this text box, specify the location of the `behat.phar` archive or the folder with the Behat executable file. Behat does not necessarily have to be installed under the current project root. You can type the path manually or click the ... button and choose the relevant location in the dialog box that opens. – Download behat.phar from https://github.com/downloads/Behat/Behat/behat.phar: click this link to have PhpStorm download the `behat.phar` archive. This approach works only if you have already an account on GitHub and you have configured access to this account in PhpStorm, see Registering GitHub Account in PhpStorm . – Behat version: this read-only field shows the version of the specified Behat installation. PhpStorm detects the version when you click the Refresh icon . The default value is Not installed
Test Runner	In this area, appoint the configuration `.yaml` file to use for launching and executing scenarios. By default, Behat looks for a `behat.yaml` configuration file in the project root folder or in the `config` folder. You can appoint a custom configuration file. – Default configuration file: select this check box to specify your own `.yaml` configuration file. Ths file will be later used as default in all Behat run/debug configurations. In the text box, specify the location of the configuration file to use. Type the path manually or click the ... button and choose the file in the dialog box that opens. Clear the check box to have Behat use the `behat.yaml` configuration file from the project root folder or from the `config` folder. If no such file is found, test execution fails, therefore it may be more reliable to specify the configuration file explicitly.

Code Sniffer

On this page:

- Code Sniffer Page
- Code Sniffer Dialog

Code Sniffer Page

File | Settings | Languages and Frameworks | PHP | Code Sniffer for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Code Sniffer for OS X

`Ctrl+Alt+S`

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, choose the Code Sniffer script to use.

Item	Description
Configuration	From this drop-down list, choose the script to use: – To use the script associated with a specific remote PHP interpreter, choose the name of this interpreter. – To use the script associated with the default project interpreter, that is, the one chosen on the PHP page of the Settings dialog box, choose By default project interpreter. – To use a local script, choose Local. In this case the local Code Sniffer will be executed no matter which PHP interpreter - local or remote - is used in the project. Note that there can be only one Local configuration for Code Sniffer because PhpStorm runs a script (`phpcs.bat` for Windows or `phpcs` for Linux) which contains a path to a PHP engine.
Ignored files	This area displays a list of files that Code Sniffer skips. PhpStorm suggests adding a new file to the list during inspection when waiting for response from the Code Sniffer exceeds the limit specified in the Tool process timeout field. This is done to prevent slowing down processing. For each file, PhpStorm displays its name and location. – To delete a file from the list and have Code Sniffer process it again, select the file and click the Remove file button —. – To remove all the files from the list, click the Clean the list button ✕.

Code Sniffer Dialog

The dialog box opens when you click ... next to the Configuration drop-down list on the Code Sniffer page.

Use this dialog box to configure local Code Sniffer scripts, scripts associated with remote PHP interpreters, see Using PHP Code Sniffer Tool , and configure Code Sniffer's behaviour.

The left-hand pane of the dialog box shows all the configured Code Sniffer scripts, one of them is of the type Local, and others are named after the remote PHP interpreters with which the scripts are associated. When you select a configuration, the right-hand pane shows its details.

- To configure or edit the Local script, select Local and specify the location of `phpcs.bat` or `phpcs` in the PHP Code Sniffer path field.
- To configure a new script associated with a remote PHP interpreter:
  1. Click + on the toolbar.
  2. In the Code Sniffer by Remote Interpreter dialog box that opens, choose the remote PHP interpreter to use the associated script from. If the list does not contain a relevant interpreter, click ... and configure a remote interpreter in the Interpreters dialog box as described in Configuring Remote PHP Interpreters .
     When you click OK, PhpStorm brings you back to the Code Sniffer dialog box where the new Code Sniffer configuration is added to the list and the right-hand pane shows the chosen remote PHP interpreter, the path to the Code Sniffer associated with it, and the advanced PHP Code Sniffer options.

Item	Description
PHP Code Sniffer (phpcs) Path	In this text box, specify the location of the Code Sniffer utility `phpcs` or `phpcs.bat` .If the script is associated with a PHP interpreter, PhpStorm detects the path to it and fills in the field automatically but you can edit it if necessary. In either case, type the path manually or click the Browse button ... and select the path in the dialog box, that opens. To check that the specified path to `phpcs.bat` or `phpcs` ensures interaction between PhpStorm and Code Sniffer, that is, the tool can be launched from PhpStorm and PhpStorm will receive problem reports from it, click the Validate button. This validation is equal to running the `phpcs --version` command. If validation passes successfully, PhpStorm displays the information on the detected Code Sniffer version.
Interpreter	The field shows the chosen PHP interpreter to use the Code Sniffer from.
Maximum number of messages per file	In this text box, set the upper limit for the total number of messages to be reported for a file. All the messages above this limit will be rejected. PhpStorm will display the following warning right in the code: `Too many PHP Code Sniffer messages` and suggest adding the file to the Ignored files list.
Tool process timeout	In this text box, specify how long you want PhpStorm to wait for a result from PHP Code Sniffer, whereupon the process is killed to prevent excessive CPU and memory usage. This gives you the capability to fine tune the PHP Code Sniffer process behavior depending on the configuration of your computer and the rule sets used.

Composer

File | Settings | Languages and Frameworks | PHP | Composer for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Composer for OS X

`Ctrl+Alt+S`

The page is available only when the PHP and the Command Line Tool Support plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, enable the use of Composer Dependency Manager through the dedicated user interface by specifying the paths to the PHP engine installation and to the `composer.phar` archive to look for Composer commands in.

When the file analyses is completed, PhpStorm returns to the Command Line Tools page of the Settings dialog bos where the specified file is added to the list of command line tools available in PhpStorm.

Item	Description
Path to PHP executable	In this field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.
Path to composer.phar	In this text box, specify the location of the `composer.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
Click here to download from getcomposer.org	Click this link to have PhpStorm download `composer.phar` from the official storage and specify the folder to store the archive in. This instance of Composer will be available in the current project only. To use it in the command line mode, configure it as a command line tool .
Path to composer.json	In this text box, specify the location of the `composer.json` for listing descriptions of the project dependencies. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
Add packages as libraries	– Select this check box to have PhpStorm mark Composer packages added to the current project as libraries.. In the context of PhpStorm, any file marked as library is automatically write-protected. Library files are indexed for the only purpose which is enhancing coding assistance (i.e. code completion, syntax highlighting, navigation, and documentation lookup). Library files are not to be edited in the scope of the project where they are declared as library. By default, the Add packages as libraries check box is selected, which automatically blocks the possibility to edit them in the current project. – Clear the check box if you want to use Composer packages in your development and at the same time update them yourself within the scope of the same project.

Mess Detector

On this page:

- Mess Detector Page
- Mess Detector Dialog

Mess Detector Page

File | Settings | Languages and Frameworks | PHP | Mess Detector for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Mess Detector for OS X

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, choose the Mess Detector script to use.

Item	Description
Configuration	From this drop-down list, choose the script to use: – To use the script associated with a specific remote PHP interpreter, choose the name of this interpreter. – To use the script associated with the default project interpreter, that is, the one chosen on the PHP page of the Settings dialog box, choose By default project interpreter. – To use a local script, choose Local. In this case the local Mess Detector will be executed no matter which PHP interpreter - local or remote - is used in the project. Note that there can be only one Local configuration for Mess Detector because PhpStorm runs a script (`phpmd.bat` for Windows or `phpmd` for Linux) which contains a path to a PHP engine.
Ignored files	This area displays a list of files that Mess Detector skips. PhpStorm suggests adding a new file to the list during inspection when waiting for response from the Mess Detector exceeds the limit specified in the Tool process timeout field. This is done to prevent slowing down processing. For each file, PhpStorm displays its name and location. – To delete a file from the list and have Mess Detector process it again, select the file and click the Remove file button —. – To remove all the files from the list, click the Clean the list button ✖.

Mess Detector Dialog

The dialog box opens when you click ... next to the Configuration drop-down list on the Mess Detector page.

Use this dialog box to configure local Mess Detector scripts, scripts associated with remote PHP interpreters, see Using PHP Mess Detector , and configure Mess Detector's behaviour.

The left-hand pane of the dialog box shows all the configured Mess Detector scripts, one of them is of the type Local, and others are named after the remote PHP interpreters with which the scripts are associated. When you select a configuration, the right-hand pane shows its details.

- To configure or edit the Local script, select Local and specify the location of `phpmd.bat` or `phpmd` in the PHP Mess Detector path field.
- To configure a new script associated with a remote PHP interpreter:
  1. Click + on the toolbar.
  2. In the Mess Detector by Remote Interpreter dialog box that opens, choose the remote PHP interpreter to use the associated script from. If the list does not contain a relevant interpreter, click ... and configure a remote interpreter in the Interpreters dialog box as described in Configuring Remote PHP Interpreters .
     When you click OK, PhpStorm brings you back to the Mess Detector dialog box where the new Mess Detector configuration is added to the list and the right-hand pane shows the chosen remote PHP interpreter, the path to the Mess Detector associated with it, and the advanced PHP Mess Detector options.

Item	Description
PHP Mess Detector (phpmd) Path	In this text box, specify the location of the Mess Detector utility `phpmd` or `phpmd.bat` .If the script is associated with a PHP interpreter, PhpStorm detects the path to it and fills in the field automatically but you can edit it if necessary. In either case, type the path manually or click the Browse button ... and select the path in the dialog box, that opens. To check that the specified path to `phpmd.bat` or `phpmd` ensures interaction between PhpStorm and Mess Detector, that is, the tool can be launched from PhpStorm and PhpStorm will receive problem reports from it, click the Validate button. This validation is equal to running the `phpmd --version` command. If validation passes successfully, PhpStorm displays the information on the detected Mess Detector version.
Interpreter	The field shows the chosen PHP interpreter to use the Mess Detector from.
Maximum number of messages per file	In this text box, set the upper limit for the total number of messages to be reported for a file. All the messages above this limit will be rejected. PhpStorm will display the following warning right in the code: `Too many PHP Mess Detector messages` and suggest adding the file to the Ignored files list.
Tool process timeout	In this text box, specify how long you want PhpStorm to wait for a result from PHP Mess Detector, whereupon the process is killed to prevent excessive CPU and memory usage. This gives you the capability to fine tune the PHP Mess Detector process behavior depending on the configuration of your computer and the rule sets used.

Phing

File | Settings | Phing for Windows and Linux

PhpStorm | Preferences | Phing for OS X

The page is available only when the Phing Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

On this page, enable Phing integration in the current project.

Item	Description
Path to Phing executable	In this text box, specify the location of the `phing.bat` file. Type the path manually or click the Browse button ... and choose the file location in the dialog box that opens.

PHPUnit

File | Settings | Languages and Frameworks | PHP | PHPUnit for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | PHPUnit for OS X

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to configure the way to launch unit tests of your PHP applications.

Item	Description

PHPUnit Library	In this area, specify where to take the PHPUnit test framework. The available options are:
	– Load from include path: choose this option to have PhpStorm run the PHPUnit from PEAR configured as an include path on the PHP page of the Settings / Preferences Dialog .
	– Use custom loader: choose this option to run the PHPUnit instance installed by the Composer Dependency Manager . The package is retrieved and loaded by the dedicated `autoload.php` file located in the `Vendor` folder. Specify the location of the `autoload.php` file in the Path to script text box. Type the path manually or click the Browse button ... and select the file in the dialog box that opens,
	– Path to phpunit.phar: choose this option to run PHPUnit from the `.phar` archive.
	  – If you already have the `phpunit.phar` archive on your computer, specify its location in the Path to phpunit.phar text box. Type the path manually or click the Browse button ... and select the file in the dialog box that opens,
	  – If you have no `.phar` archive on your computer yet, click the Download phpunit.phar ... link to have PhpStorm download it automatically.
	  In either case, PhpStorm will load the archive before test execution.

Test Runner	In this area, appoint the configuration `XML` file to use for launching and executing scenarios. By default, PHPUnit looks for a `phpunit.xml` configuration file in the project root folder or in the `config` folder. You can appoint a custom configuration file. – Default configuration file: select this check box to specify your own `XML` configuration file. Ths file will be later used as default in all PHPUnit run/debug configurations. In the text box, specify the location of the configuration file to use. Type the path manually or click the ... button and choose the file in the dialog box that opens. Clear the check box to have PHPUnit use the `phpunit.xml` configuration file from the project root folder or from the `config` folder. If no such file is found, test execution fails, therefore it may be more reliable to specify the configuration file explicitly. – Default bootstrap file: select this check box to have PHP script always executed before launching tests. In the text box, specify the location of the script. Type the path manually or click the Browse button ... and select the desired folder in the dialog that opens .

Blade

File | Settings | Languages and Frameworks | PHP | Blade for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Blade for OS X

`Ctrl+Alt+S`

This page is available only when the PHP and Blade plugins are installed and enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

The page consists of the following tabs:

- Text Tags
- Directives

Text Tags

In this tab, specify the delimiters to use in Blade templates. According to these settings, PhpStorm recognizes templates and provides error highlighting and code completion for them.

The fields in the tab show the opening and closing characters for raw tags , content tags , and escaped tags .

The fields are filled in with the default values in compliance with Blade Templates 5.1 . If you are using an earlier version, you can specify the relevant custom delimiters and PhpStorm will provide coding assistance according to the new rules.

Directives

In this tab, manage Blade directives for use in PhpStorm. The tab lists all the currently available Blade directives, for those that have parameters, the prefixes and suffixes are also shown. When you start, the list contains only predefined directives. You can edit these directives as well as create custom ones.

Item	Description
Directives	The tab lists all the currently available Blade directives, for those that have parameters, the prefixes and suffixes are also shown. When you start, the list contains only predefined directives. You can edit these directives as well as create custom ones. – Add directive (+) click this button to define a new directive. Specify the directive's name in the Name text box. If the new directives requires a prefix and a suffix, select the Has parameter check box and type the prefix and suffix to use in the Prefix and Suffix text boxes respectively. PhpStorm will automatically enclose the prefix and suffix in opening and closing brackets and quotes and add a colon separator `:` so the parameters will look as follows: `("<prefix>:<suffix>")` . To edit an existing directive, select it in the list and change the values in the text boxes below. To restore the original definition, click the Reset to defaults button . – Remove directive(s) (—) click this button to remove the selected directive from the list. – Reset to defaults ( ) click this button to restore the original definition of the selected directive.
Name	In this text box, specify the name of a new directive or edit the name of the selected one.
Has parameter	Select this check bo to specify a prefix and a suffix for a new directive or for the one selected in the list. When the check box is selected, the Prefix and Suffix text boxes are available for editing. Specify the required directive parameters. If the new directives requires a prefix and a suffix, select the Has parameter check box and type the prefix and suffix to use in the Prefix and Suffix text boxes respectively. PhpStorm will automatically enclose the prefix and suffix in opening and closing brackets and quotes and add a colon separator `:` so the parameters will look as follows: `("<prefix>:<suffix>")` .

Drupal

File | Settings | Languages and Frameworks| PHP | Drupal for Windows and Linux

for OS X

`Ctrl+Alt+S`

Use this page to enable and configure integration with Drupal in a project.

Item	Description
Enable Drupal integration	Select this check box to have the Drupal functionality applied to your sources.
Drupal installation path	In this text box, specify the root folder of the Drupal installation. Type the path manually or click the Browse button and select the relevant folder in the dialog box that opens.
Set up PHP | Include paths	– Select this check box to have Drupal include paths automatically configured for the project. The following paths will be added to the Include Paths list on the PHP page of the Settings dialog box: 1. `<drupal installation root>/includes` 2. `<drupal installation root>/modules` 3. `<drupal installation root>/sites/all/modules` The include paths are added to the list after you leave the dialog box. – Clear the check box to configure the include paths manually.
Version	From this drop-down list, choose the version of Drupal to use, the supported versions are 6, 7, and 8 .

Google App Engine for PHP

File | Settings | Languages&Frameworks | Node.js and NPM for Windows and Linux

PhpStorm | Preferences | Languages&Frameworks | Node.js and NPM for OS X

Ctrl+Alt+S

In this dialog box, activate and configure the support of the Google App Engine for PHP in an existing PhpStorm project. For more details, see Preparing to Develop a Google App for PHP Application .

Item	Description
Enable Google App Engine for PHP support	– When this check box is selected, PhpStorm provides assistance in developing PHP applications intended for running in the Google PHP Runtime Environment . This support includes coding assistance and the possibility to run and debug such PHP applications on the local Google development server. – When the check box is cleared, no assistance in developing PHP applications for running in the Google runtime environment is provided and all the controls on the page are disabled.
SDK directory	In this field, specify the folder where the Google App Engine for PHP SDK is stored.
Python executable	In this field, specify the path to the Python executable file, version 2.7 is required.
App Engine Account Settings	In this area, choose the way to authenticate to the development server, the available options are: – Use passwordless login via OAuth2: choose this option to use the OAuth 2.0 protocol . To save the token achieved through the Google Developers Console , clear the Do not save token check box. – Log in with email and password: choose this option to use your Gmail address and password.

Joomla! Support

File | Settings | Languages and Frameworks| PHP | Joomla! for Windows and Linux

for OS X

Ctrl+Alt+S

Use this page to enable and configure integration with Joomla! in a project.

Item	Description
Enable Joomla! integration	Select this check box to have the Joomla! functionality applied to your sources.
Joomla! installation path	In this text box, specify the root folder of the Joomla! installation. Type the path manually or click the Browse button ... and select the relevant folder in the dialog box that opens.

Smarty

File | Settings | Languages and Frameworks | PHP | Smarty for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Smarty for OS X

`Ctrl+Alt+S`

The page is available only when the PHP plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this dialog box to specify the delimiters to enclose Smarty tags in.

Item	Description
Smarty left delimiter	In this text box, specify the desired opening delimiter. By default, the field shows the opening curly brace `{` .
Smarty right delimiter	In this text box, specify the desired closing delimiter. By default, the field shows the closing curly brace `}` .
Use Smarty 3 whitespace policy	When this check box is selected, delimiters followed by whitespace will not be parsed as template tags. The check box is available when Smarty 3.0 or higher is used.

WordPress
PhpStorm

Schemas and DTDs

File | Settings | Languages and Frameworks | Schemas and DTDs for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Schemas and DTDs for OS X

Ctrl+Alt+S

The settings on this page define how your XML, HTML and XHTML files are validated.

- External Schemas and DTDs
- Ignored Schemas and DTDs

External Schemas and DTDs

Local XML schema (XSD) and DTD files that are used to validate your XML files are listed in this section.

Each entry is a mapping of a URI that may be referenced in your XML file onto an appropriate local schema or DTD file. Use +, − and to create, remove and edit the mappings. See also, Referencing XML Schemas and DTDs .

Item	Keyboard Shortcut	Description
URI		A URL of an XML schema (XSD) or DTD file (e.g. `http://www.example.org/xsds/example.xsd`) or a namespace URI (e.g. `http://www.example.org` , `urn:jboss:domain:1.0`). Readonly. If an XML file references the specified URI, it's validated according to a local file whose path is shown in the Location columns.
Location		Path to corresponding local XSD or DTD file (readonly).
Project		If a check box is not selected, a mapping is available in all of your projects. Select the mappings that you want to be available only in the current project. The selected mappings are stored in `.idea/misc.xml` or the project `.ipr` file. These files are normally shared between development team members through version control.
+	Alt+Insert	Use this icon or shortcut to open the Map External Resource Dialog to define a new mapping between a URI and a local file.
−	Alt+Delete	Use this icon or shortcut to remove the selected mappings from the list.
	Enter	Use this icon or shortcut to edit the selected mapping. (The Map External Resource Dialog dialog will open.)

Ignored Schemas and DTDs

URIs for ignored schemas and DTDs are listed in this section. (If an XML file references a URI listed in this section, PhpStorm ignores that URI and doesn't mark it as an error in the editor.) Use +, − and to add, remove and edit the URIs. See also, Referencing XML Schemas and DTDs .

Item	Keyboard Shortcut	Description
+	Alt+Insert	Use this icon or shortcut to add a new URI to the list.
	Enter	Use this icon or shortcut to edit the selected URI.
−	Alt+Delete	Use this icon or shortcut to remove the selected URIs from the list. The corresponding URIs become available for mapping to local XSD or DTD files.

Default XML Schemas

File | Settings | Languages and Frameworks | Schemas and DTDs | Default XML Schema for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Schemas and DTDs | Default XML Schema for OS X

`Ctrl+Alt+S`

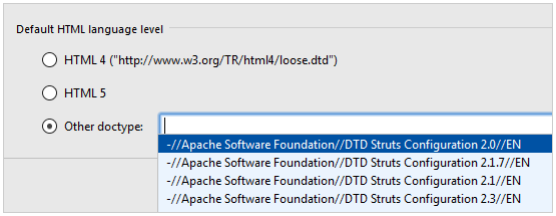
On this page:

- Default HTML Language Level
- Default XML Schema Version

Default HTML Language Level

Normally, an HTML or an XHTML file has the `<!DOCTYPE>` declaration which states the language level of the used in the source code from the file. This language level is used as a standard against which the contents of the file are validated. If an HTML or an XHTML file does not have a `<!DOCTYPE>` declaration, the contents of the file will be validated against the default standard (schema). In the Default HTML Language Level area, choose the default schema to validate HTML and XHTML files without a `<!DOCTYPE>` declaration. The available options are:

- HTML 4 or HTML 5: Choose one of these options to have files treated as HTML 4 or HTML 5 and validated against one of these standards.
- Other doctype: Choose this option to have HTML files by default validated agains a custom DTD or schema and specify the URL of the DTD or schema to be used. Note that code completion is available in this field: press `Ctrl+Space` to see the list of suggested URLs.



Default XML Schema Version

In this area, choose the XSD (XML Schema Definition) Schema to validate XML files. The available options are:

- XML Schema 1.1 See W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures and W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes for details.
- XML Schema 1.0 See XML Schema Part 1: Structures Second Edition and XML Schema Part 2: Datatypes Second Edition for details.

XML Catalog

File | Settings | Languages and Frameworks | Schemas and DTDs | XML Catalog for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Schemas and DTDs | XML Catalog for OS X

Ctrl+Alt+S

Use this dialog to configure your .properties file. This file contains lists of catalog.xml files. The catalog.xml files contain the information on how to resolve various PUBLIC and SYSTEM identifiers during XML processing.

Item	Description
Catalog property file	Use this field to specify the location of the .propeties file. Type the path in the field, or click ... and select the required file in the dialog that opens . For more detailed description of the properties in the catalog property file, see the example of Annotated CatalogManager properties file .

Compass

File | Settings | Languages and Frameworks | Compass for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | Compass for OS X

`Ctrl+Alt+S`

Use this page to integrate the Compass framework with PhpStorm and thus enable transpilation of Sass and SCSS files from Compass-specific projects into CSS. For details, see Using File Watchers .

Prerequisites

Before you start working with Compass, make sure that the Sass support plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Make sure that Ruby is installed on your computer.

For more details, see Working with Sass and SCSS in Compass Projects .

Item	Description
Enable Compass support	Select this check box to activate Compass support including the possibility to transpile Sass and SCSS files from your Compass project into CSS.
Compass executable file	In this field, specify the location of the `compass` executable file under the Ruby installation. Type the path manually, for example, `C:\Ruby200-x64\bin\compass`, or choose it from the drop-down list, or click the Browse button and choose the location of the `compass` file in the dialog box that opens.
Config path	In this field, specify the location of the project Compass configuration file `config.rb`. Type the path manually, for example, `C:\my_projects\\compass_project\config.rb`, or choose it from the drop-down list, or click the Browse button and choose the location of the `compass` file int he dialog box that opens. The Compass configuration file `config.rb` is generated during project set-up through `compass create` or `compass init` commands.

JSON Schema

File | Settings | Languages and Frameworks | JSON Schema for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JSON Schema for OS X

`Ctrl+Alt+S`

On this page, create a list of your custom JSON schemas in addition to the system schemas provided by PhpStorm and configure the scope for each schema, that is, specify the `.json` files to be validated against it. The page consists of two panes: the List of Schemas pane and the Schema Details pane.

On this page:

- List of Schemas Pane
- Schema Details Pane
- Handling Conflicts Among Scopes of Schemas

List of Schemas Pane

The central pane of the page shows a list of custom JSON schemas against which `.json` files in the current project will be validated. Based on a schema, PhpStorm checks the structure of `.json` configuration files, reports errors (for example, informs you about missing mandatory properties), provides code completion and documentation look-up.

The pane shows only custom schemas that you have previously downloaded or created yourself, in either case, a custom schema must meet the JSON schema standards and must be located under the project root.

System schemas that PhpStorm provides for all supported frameworks and technologies are not shown in the list.

- To add a schema to the list, click + on the toolbar of the pane and select the relevant schema file under the project root in the dialog box that opens.
- To remove a schema from the list, select it and click — on the toolbar of the pane.
- To configure the scope of a schema, select the schema and specify the files to be validated against it in the right-hand pane of the page.

Schema Details Pane

The pane shows the details of the schema selected in the List of Schemas pane: the name of the selected schema, the `.json` file that implements it, and a list of files and folders that are validated against the schema.

The list contains the names of specific files, the names of entire directories, and filename patterns. You do not need to specify full paths to files and folders, PhpStorm searches for files and folders with the specified names and the search is restricted to the the current project.

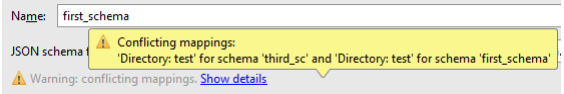
Based on the list, PhpStorm internally creates a list of files to be validated. Each file is included in the list only once: if a file with the specified name is stored in a directory from the list or its name matches a pattern, the file is still validated only once.

- To add new files or folders to the list, click + on the toolbar of the pane and specify the file, folder, or a file pattern in the Add JSON Schema Mapping Dialog that opens.
- To remove an item from the list, select it and click — on the toolbar of the pane.

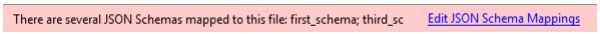
Handling Conflicts Among Scopes of Schemas

A conflict arises when a file, or a folder, or a pattern belongs to the scopes of two or more schemas. PhpStorm analyzes scopes in two modes:

- Static Analysis detects conflicts in scopes of custom schemas. If a conflict is detected, PhpStorm displays a warning in the Schema Details pane. To view the overlapping scopes, click the Show details link. PhpStorm shows a pop-up message where the conflicting scopes and schemas are listed:



- Dynamic Analysis detects conflicts in scopes of both system and custom schemas. This type of analysis is started when you open a file that belongs to a certain scope. If a conflict is detected, PhpStorm displays a warning at the top of the editor tab:



Click the link to open the JSON Schema page and edit the scope of the conflicting custom schema. Note that you cannot edit the scope of system schemas.

Add JSON Schema Mapping Dialog

File | Settings | Languages and Frameworks | JSON Schema for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JSON Schema for OS X

The dialog box opens when you select a `.json` schema in the list on the JSON Schema page and click + in the right-hand pane. In this dialog, specify the `.json` files to be validated against the selected schema.

Item	Description
Files Under	Choose this option to add files stored in a specific folder name to validation. Type the name of the folder manually or click ... and choose the folder in the dialog box that opens.
Filename pattern	Choose this option to have all files with the names that match a pattern validated.
File	Choose this option to add files with a specific name to validation. Type the filename manually or click ... and choose the file in the dialog box that opens.

Node.js and NPM

File | Settings | Languages&Frameworks | Node.js and NPM for Windows and Linux

PhpStorm | Preferences | Languages&Frameworks | Node.js and NPM for OS X

`Ctrl+Alt+S`

This page appears in the Settings dialog box, when the Node.js plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

Item	Description
Node interpreter	In this field, specify the local Node.js interpreter to use. Choose the interpreter from the drop-down list or click ... and choose the interpreter in the dialog box that opens. The term local Node.js interpreter denotes a Node.js installation on your computer. The term remote Node.js interpreter denotes a Node.js installation on a remote host or in a virtual environment set up in a Vagrant instance. On the Node.js and NPM page, you can specify only local interpreters. Remote interpreters are configured in the Configure Node.js Remote Interpreter Dialog dialog which can be accessed only from the Run/Debug Configuration: Node JS dialog. See Configuring Node.js Interpreters for details.
Version	This read-only field shows the current version of the runtime environment.
Code Assistance	When developing a Node.js application it can be convenient to have code completion, reference resolution, validation, and debugging capabilities for Node core modules (`fs` , `path` , `http` , etc.). However, these modules are compiled into the Node.js binary. PhpStorm provides the ability to configure these sources as a JavaScript library and associate it with your project. – If the Node.js core module sources are not set up, PhpStorm displays a notification Node.js Core Library is not enabled with an Enable button. Click this button to have PhpStorm configure Node.js Core sources automatically. When the configuration is completed, PhpStorm displays information about the currently configured version, the notification Node.js Core Library is enabled, and adds two buttons: the Disable button and the Usage scope button. – If the library is set up, PhpStorm displays information about the currently configured version, the notification Node.js Core Library is enabled, and adds two buttons: the Disable button and the Usage scope button. – Click the Disable button to discard the configuration of the Node.js Core libraries in the current project. – Click the Usage scope button to associate the desired directories with libraries. If necessary, configure the scope in which the Node.js Core sources are treated as libraries. Click the Usage scope button, and in the Usage Scope dialog box that opens, click the desired directories, and from the drop-down list select the newly configured Node.js core module sources library.
Packages	A number of tools are started through Node.js, for example, the CoffeeScript, TypeScript, and Less transpilers, YUI, UglifyJS, and Closure compressors, Karma test runner, Grunt task runner, etc. The Node Package Manager (npm) is the easiest way to install these tools, the more so that you have to install Node.js anyway. The Packages area shows a list of all the NPM-dependent packages that are currently installed on your computer. – Package: this read-only field shows the name of a package, exactly as it should be referenced if you were installing it in the command line mode. – Version: this read-only field shows the version of the package installed on your computer. – Latest: this read-only field shows the latest released version of the package. If a package is not up-to-date, it is marked with a blue arrow. – Click + to have a new package installed. In the Available Packages dialog box that opens, select the relevant package. To have the package installed globally, select the Options check box and type `-g` in the Options text box. Global installation makes the package available at the PhpStorm level so it can be used in any PhpStorm project. Click Install Package when ready. – Click – to have the selected package removed. – Click ↑ to have the current version of the selected package replaced with the latest released version. The button is enabled only when the selected project is not up-to-date.

TypeScript

File | Settings | Languages & Frameworks | TypeScript for Windows and Linux

PhpStorm | Preferences | Languages & Frameworks | TypeScript for OS X

`Ctrl+Alt+S`

In this dialog, activate, deactivate, and configure the behaviour of the built-in TypeScript compiler that transpiles TypeScript code into JavaScript so the code can be executed in a browser and generates sources maps that ensure correspondence between the lines in the generated JavaScript code and the lines in the original TypeScript code.

Item	Description
Node interpreter	In this field, specify the location of the Node.js executable file. In most cases, PhpStorm detects the Node.js executable and fills in the field automatically.
TypeScript version	In this area, specify the version of the compiler to use. PhpStorm displays the default bundled version. To use a custom version of the compiler: 1. Download the `typescriptServices.js`, `lib.d.ts`, and `lib.es6.d.ts` files from https://github.com/Microsoft/TypeScript/ . 2. Click Edit in the Compiler version area. 3. In the Configure TypeScript Compiler dialog box that opens, choose Custom directory and specify the folder where the downloaded files are stored.
Use TypeScript Service	Select this check box to get native support from the TypeScript Language Service according to the up-to-date specifications. In this case syntax and error highlighting is performed based on the annotations retrieved from the TypeScript Language Service while code completion lists contain both suggestions from the TypeScript Language Service and suggestions calculated by PhpStorm itself.
Enable TypeScript Compiler	Select this check box to activate the built-in compiler. When the check box is selected, syntax highlighting and code completion are provided based only on the data from the built-in Typescript compiler. After you select this check box, the fields below on the page become active and you can configure the behaviour of the compiler.
Track changes	– When this check box is selected, the built-in compiler "wakes up" upon any change to a TypeScript file. – When this check box is cleared, the built-in compiler ignores changes to TypeScript files. To re-activate the compiler, open the TypeScript Compiler Tool Window (View | Tool Windows | TypeScript Compiler), and click the Compile All button on the toolbar. If you have not opened the TypeScript Compiler Tool Window yet and it is not available from the View menu, choose Help | Find Action, then find and launch the TypeScript Compile All action from the list.
Scope	From this drop-down list, choose the scope to apply the compiler in. The available options are: – Project Files: all the files within the project content roots (see Content Root and Configuring Content Roots). – Project Production Files: all the files within the project content roots excluding test sources. – Project Test Files: all the files within the project test source roots. – Open Files: all the files that are currently opened in the editor. VCS Scopes: these scopes are only available if your project is under version control . – Changed Files: all changed files, that is, all files associated with all existing changelists . – Default: all the files associated with the changelist `Default` . Alternatively, click the Browse button and configure a custom scope in the Scopes dialog box that opens. For more details on scopes, see the pages Scopes and Scopes dialog .

Set options manually	Choose this option to configure the behaviour of the built-in compiler manually:
	– In Command line options field, specify the command line options to be passed to the compiler. See the list of acceptable options at TSC arguments . Note that, the `-w` or `--watch` option (Watch input files) is irrelevant.
	– Select the Generate source maps check box to generate source maps that set correspondence between lines in your TypeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.
	– Select the Compile main file only check box to have PhpStorm compile only a specific file and the files that are referenced from it and ignore all the other files in the project. This may be helpful if you have a dedicated `main.ts` file which only references other files.
	– Select the Use output path check box to have the built-in compiler store the generated JavaScript files and source map in a custom folder. Specify the path to this folder explicitly or use one of the listed available macros in the format: `$<macros_name>$` . The available macros are:
	  – `FileDir` : the path to the folder where the file is stored.
	  – `FileRelativeDir` : the path to the file directory relative to the project root.
	  – `FileDirRelativeToProjectRoot` : the path to the file directory relative to the project root.
	  – `ModuleFileDir` : the path to the project root folder.
	  – `SourcePath` : the path to the source folder under the content root to which the file belongs, see Configuring Content Roots and Configuring Folders Within a Content Root .
	  – `FileDirRelativeToSourcePath` : the path to the file relative to the source folder under the content root to the file belongs.

Use tsconfig.json	Choose this option to have the built-in compiler analyze the code according to the settings specified in the `tsconfig.json` file .

TSLint

File | Settings | Languages and Frameworks | JavaScript | Code Quality Tools | TSLint for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | JavaScript | Code Quality Tools | TSLint for OS X

`Ctrl+Alt+S`

Use this page to activate integration with the TSLint code verifier and configure its behaviour and sensitivity.

Item	Description
Enable	Select this check box to have TSLint applied to verify the code in the current project. After that the other controls in the page are enabled.
Node Interpreter	In this field, specify the NodeJS installation home. Type the path to the NodeJS executable file manually, or click the ... button and select the location in the dialog box, that opens. If you have appointed one of the installations as default , the field displays the path to its executable file.
TSLint Package	In this field, specify the location of the `tslint` package installed globally or in the current project, see Using TSLint Code Quality Tool .
Configuration File	In this area, appoint the configuration to use. By default, PhpStorm first looks for a `tslint.json` configuration file. PhpStorm starts the search from the folder where the file to be checked is stored, then searches in the parent folder, and so on until reaches the project root. If no `tslint.json` file is found, TSLint uses its default embedded configuration file. Accordingly, you have to define the configuration to apply either in a `tslint.json` configuration file, or in a custom JSON configuration file, or rely on the default embedded configuration. – To have PhpStorm look for a `tslint.json` file, choose the Search for tslint.json option. If no `tslint.json` file is found, the default embedded configuration file will be used. – To use a custom file, choose the Configuration File option and specify the location fo the file in the Path field. Choose the path from the drop-down list, or type it manually, or click the ... button and select the relevant file from the dialog box that opens.
Additional Rules Directory	In this field, specify the location of the files with additional code verification rules. These rules will be applied after the rules from `tslint.json` or the above specified custom configuration file and accordingly will override them.

SQL Dialects

File | Settings | Languages and Frameworks | SQL Dialects for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | SQL Dialects for OS X

This page lets you specify which SQL dialects are used in your project.

A dialect may be set at the project level, and also at the level of a directory or a file.

If a dialect is not specified explicitly, it is inherited from a higher hierarchical level.

The dialects specified explicitly are shown in black. The inherited values are shown in gray italic.

Note For this page to be available, the Database Tools and SQL plugin must be enabled. See Enabling and Disabling Plugins .

Item	Description
File/Directory	This column shows the hierarchy of files and folders in the project. Each row corresponds to a directory or a file.
SQL Dialect	Specify the SQL dialect to be used. Click the cell of interest and select the necessary option from the list. In addition to particular dialects, you can select: – Clear. This will clear the corresponding cell. As a result, an option from a higher hierarchical level will be inherited. – <Generic>. This means that no particular dialect is specified. As a result, basic SQL92-based coding assistance is provided including completion and highlighting for SQL keywords, and table and column names. Syntax error highlighting is not available. So the file contents are always shown as syntactically correct. Also, code reformatting isn't possible.

XSLT

File | Settings | Languages and Frameworks | XSLT for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | XSLT for OS X

Ctrl+Alt+S

This page appears if XPath View+XSLT Support plugin is enabled. Use this page to configure XSLT support at the PhpStorm level.

Item	Description
Show Associated Files in Project View	Select this check box to have XML files associated with XSLT stylesheets displayed in the project view. Associated XML files are displayed below the corresponding stylesheets.

XSLT File Associations

File | Settings | Languages and Frameworks | XSLT File Associations for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | XSLT File Associations for OS X

Ctrl+Alt+S

Use this page to associate an XSLT stylesheet file with XML files. This is necessary to enable error highlighting and enhanced completion for element and attribute names in XSLT node-selections.

Item	Description
Project XSLT Files	The pane shows all the XSLT files in a project tree view, grouped by modules and their content roots.
Associated Files	The pane shows all the XML file associated with the selected XSLT file.
+	Click this button to select the XML file that should be associated with the selected XSLT file. See the description of the dialog that opens . The button is available only when an XSLT file is selected in the Project XSLT Files pane.
−	Click this button to cancel the association between the XML file selected in the Associated Files pane and the XSLT file selected in the Project XSLT Files pane.

Dart

The dialog box opens when you click the Dart Settings button on the toolbar. Use the dialog box to configure the Dart SDK and the Dartium browser. The SDK is not bundled with PhpStorm and has to be downloaded manually. For information on the available versions of Dart SDK and for installation instructions, please refer to http://www.dartlang.org/downloads.html .

Item	Description
Enable Dart support in project <project name>	– When this check box is selected, PhpStorm provides assistance in coding, testing, running, and debugging Dart applications and enables you to configure the Dart SDK and the Dartium browser. – When the check box is cleared, no assistance in developing Dart applications is provided and all the controls on the page are disabled.
Dart SDK Path	In this text box, specify the location of the downloaded Dart SDK. Type the path manually or click the Browse button and choose the path in the dialog box that opens. If PhpStorm recognizes the Dart SDK correctly, it displays the revision number of the detected SDK in the Version read-only field.
Version	In this read-only field, PhpStorm shows the revision number of the detected Dart SDK, provided that the SDK is recognized correctly.
Check SDK update	When this check box is selected, PhpStorm checks whether the specified above version of SDK is the latest one. If a newer version of SDK is available, PhpStorm displays a pane at the top of the active editor informing you that a newer SDK version has been released. Do one of the following: – Click the Download SDK link and download it from the site. – If you have already downloaded the latest SDK version, click the Dart Settings link to switch to the Dart page of the Settings dialog box and specify the new SDK location. From the drop-down list, choose the release types of SDK to look in, the available options are: – Stable channel – Stable and Dev channels
Dartium Path	In this text box, specify the location of the Dartium executable file. In the Dartium browser , you can run Dart code without previously transpiling it into JavaScript. Type the path manually or click the Browse button and choose the path in the dialog box that opens. If the SDK was downloaded within the Dart Editor Bundle, the path to Dartium is also detected and filled in automatically. Learn more about Dart development tools at the Dart Download Page .
Start Dartium in checked mode	
Custom package roots	

Tools

File | Settings | Tools for Windows and Linux

for OS X

Ctrl+Alt+S

When you select the Tools category in the left-hand pane, its main subcategories are listed in the right-hand part of the dialog.

- Web Browsers
- File Watchers
- External Tools
- Terminal
- Database
- SSH Terminal
- Command Line Tool Support
- Diagrams
- Diff & Merge
- Docker Machine
- Remote SSH External Tools
- Server Certificates
- Settings Repository
- Startup Tasks
- Tasks
- Vagrant
- WordPress
- XPath Viewer

Web Browsers

File | Settings | Tools | Web Browsers for Windows and Linux

PhpStorm | Preferences | Tools | Web Browsers for OS X

`Ctrl+Alt+S`

On this page

- Integrate installations of Web browsers with PhpStorm, activate or deactivate launching Web browsers from PhpStorm .
- Specify whether a browser will be launched by running its executable file or through the default system command .
- Appoint the default PhpStorm browser in which PhpStorm will open HTML files upon request by default, that is, when no browser is specified explicitly .

Browsers

In this section, specify which browsers will be available for previewing HTML output. The section shows a predefined list of browsers, possibly extended with previously configured custom browser installations. Each browser is presented as a separate table row.

PhpStorm is shipped with a predefined list of most popular browsers which you may like to install and use. The items are added to the list in advanced and are not based on the information on actually installed browsers. PhpStorm presumes that you install browsers according to a standard procedure. Based on this assumption, each browser in this predefined list is assigned an alias which stands for the path to its executable file, as PhpStorm supposes it to be. If in your actual browser installation the path to the executable file is different, you need to specify it explicitly in the Path field.
In addition to the predefined browsers, you can configure as many custom browser installations as you need using the controls on the toolbar.

Column	Description
Active	Select this check box to enable the use the respective browser from PhpStorm. The browser will be added to the context menu of the Open in Browse menu item and its icon will be displayed in the Browsers pop-up toolbar. If this check box is cleared, the corresponding browser icon will not appear in the icons toolbar or pop-up menu.
Name	In this column, specify the browser name.
Family	In this column, specify the family to which the browser belongs.
Path	In this column, specify the path to the executable. If the browser was installed according to a standard installation procedure, most likely the alias shown in the Path field points at the right location of the executable file. To specify the path explicitly, click in the Path field and choose the actual location of the executable file in the dialog box that opens. In the dialog that opens , choose the path to the executable file of the corresponding browser.

Toolbar

Item	Description
+	Click this button to add a custom browser to the list.
−	Click this button to delete the selected customer browser from the list. Note that you cannot delete the browsers from the predefined list.
	Click this button to specify a custom profile for Firefox or a browser of the Chrome family. The button is available only when Firefox and Chrome are selected. In the Firefox Settings dialog box, specify the Firefox browser profile to use for previewing output: – Path to "profiles.ini": in this text box, specify the location of the `profiles.ini` file, which determines the Firefox profile to be used. – Profile: from this drop-down list, select the desired predefined profile to use. Learn more at Firefox browser profile . In the Chrome Settings dialog box: – Command line options: In this text box, enter the command line options to launch an instance of Chrome. If you need more space, click , or press `Shift+Enter` to open the editor box. Learn more about Chrome command line options by opening chrome://flags in Chrome. – Use custom profile directory: Select this check box to define a user-specific Chrome profile to use and specify the location of the `chrome-user-data` directory, which determines the Chrome profile to be used. Learn more about Chrome profiles at Multi-profiles .
	Use these buttons to move the selected browser up or down in the list. The order of browsers is important for rendering external resources and previewing files with Web contents.
	Click this button to create a copy of the selected browser.

Default Browser

In this section, specify the default PhpStorm browser that will be used by default for rendering external resources and previewing files with Web contents. This browser will be referred to as Default in the context menu when you choose View | Open in Browser on the main menu or Open in Browser on the context menu of a file.

Item	Description
Default browser	Select the default browser from the drop-down list. The possible options are: – System default: Select this option to accept your operating system default Web browser as default for PhpStorm. – First listed: Select this option to have PhpStorm launch the first browser in the list. Change the order or browsers using the and icons on the toolbar. – Custom path: Select this option to specify another Web browser as default for PhpStorm. Type the path to the executable file of the browser or click and select the path in the dialog box that opens.
Show browser popup in the editor	– If this check box is selected, the popup window with the enabled browsers appears in the HTML files. – If this check box is not selected, this popup window does not show up, thus helping you read or edit code.

File Watchers

File | Settings | Tools | File Watchers for Windows and Linux

for OS X

Ctrl+Alt+S

The page is available when the File Watchers plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to create project File Watchers based on predefined PhpStorm File Watcher templates and thus enable transpilation in the project.

PhpStorm supports integration with various third-party transpilers that run in the background and perform the following:

- Translate Less, Sass, and SCSS source code into CSS code.
- Translate TypeScript and CoffeeScript source code into JavaScript code, possibly also creating source maps to enable debugging.
- Compress JavaScript and CSS code.

Note that PhpStorm does not contain built-in transpilers but only supports integration with the tools that you have to download and install outside PhpStorm.

In PhpStorm, these transpiler configurations are called File Watchers. For each supported transpiler, PhpStorm provides a predefined File Watcher template. Predefined File Watcher templates are available at the PhpStorm level. To run a transpiler against your project files you need to create a project-specific File Watcher based on the relevant template, at least, specify the path to the transpiler to use on your machine.

You can download a transpiler of your choice and set it up as a File Watcher. However, in this case no predefined template is available so you will have to specify all the settings manually.

To be applicable, a File Watcher must be enabled by selecting the check box next to it on the File Watchers page of the Settings dialog box, see Enabling and Disabling File Watchers . After that the File Watcher will be invoked automatically upon any changes made to the source code or upon save, depending on whether the Immediate File Synchronization check box is selected or cleared, see New Watcher Dialog .

The output of a File Watcher is stored in a separate file. The predefined templates suggest the type of the file depending on the transpiler type. By default the output file is created in the same folder as the input file when the File Watcher is invoked for the first time, whereupon this file is only updated. You can customize all these settings during File Watcher creation.

JavaScript files generated by File Watchers are excluded from code completion and refactoring.

In the Project tree view, the output file is shown under the original file which is shown as a node. This is done to improve visibility so you can easier locate necessary files.

File watchers have two dedicated code inspections:

- The File watcher available inspection is invoked in every file that is recognized as subject for applying a predefined file watcher (Sass, Less, SCSS, TypeScript, or CoffeeScript). If none of the applicable predefined File Watchers is associated with the current project, PhpStorm suggests to add one.
- The File watcher problems inspection is invoked by a running File Watcher and highlights errors specific for it.

The File Watchers page consists of two parts:

- A list of File Watchers available in the current project. To activate a File Watcher, select the check box next to it. If an error occurs while a File Watcher is running, the File Watcher is automatically disabled.
- A toolbar to manage this list.

ItemTooltip/ Description

and shortcut		
+	Add Alt+Insert	Click this button to open the Choose template pop-up list and choose the relevant type of File Watcher. After that PhpStorm opens the New Watcher dialog box for customizing the predefined File Watcher according to the settings of the current project.
	Edit Enter	Click this button to update the settings of the selected File Watcher in the Edit Watcher dialog box. The update is applied to the current project File Watcher only, it does not affect the predefined PhpStorm-level template.
−	Remove Alt+Delete	Click this button to remove the selected File Watcher. The File Watcher is no longer applied to the files in the current project. Note that this action does not affect the corresponding predefined template which is still available at the PhpStorm level.
	Up(Ctrl+Alt+Up) Down (Ctrl+Alt+Down)	Use these buttons to change the order of File Watcher in the list. This determines the order of launching File Watchers, if more than one are enabled.
	Copy	Use this button to create a copy of the selected file watcher.
	Import	Click this button to import an existing file watcher and add it to the list of available file watchers.
	Export	Click this button to export the selected watchers to `watchers.xml` file, located under the user's home.

New Watcher Dialog

File | Settings | Tools | File Watchers for Windows and Linux

PhpStorm | Preferences | Tools | File Watchers for OS X

Ctrl+Alt+S

The dialog box opens when you click the Add + or Edit button on the File Watchers page . Use the dialog box to create a project File Watcher based on a predefined PhpStorm File Watcher template or to edit an existing project File Watcher.

Each template contains the settings that are optimal for the selected transpiling tool. So in most cases, all you need is specify the path to the transpiler executable.

On this page:

- Name and Description Area
- Options Area
- Watcher Settings Area
- Examples of customizing the behaviour of a transpiler

Name and Description Area

Item	Description
Name	In this text box, type the name of the File Watcher. By default, PhpStorm suggests the name of the selected predefined template.
Description	In this text box, describe the functionality of the File Watcher. Predefined templates already contain general information regarding the accepted input and produced output. Expand or update the description, if, for example, you have two File Watcher that apply to the same source code but differ from each other in some configuration settings.

Options Area

In this area, configure the behaviour of the File Watcher.

Item	Description
Output Filters	Click this button to open the Output Filters dialog where you can manage the list of filters to distinguish the output of the File Watcher from other output. These filters make the basis for: 1. Displaying paths to the File Watcher output files as links in error and other messages and logs. When you click such link, the corresponding file is opened in the editor. 2. Error highlighting in the output files.
Immediate file synchronization	– Select this check box to have the File Watcher invoked as soon as any changes are made to the source code. – If the check box is cleared, the File Watcher will start upon save (File | Save All, ?) or when you move focus from PhpStorm (upon frame deactivation).
Track only root files	In the TypeScript file watcher, the check box has been renamed to Compile main files only. When the File Watcher is invoked on a file, PhpStorm detects all the files in which this file is included. For each of these files, in its turn, PhpStorm again detects the files into which it is included. This operation is repeated recursively until PhpStorm reaches the files that are not included anywhere within the specified scope . These files are referred to as root files (do not confuse with content roots). – When this check box is selected, the File Watcher runs only against the root files. – When the check box is cleared, the File Watcher runs against the file from which it is invoked and against all the files in which this file is included recursively within the specified scope. This option is available only for Babel, Closure Compiler, Compass, Jade, Less, Sass/ SCSS, Stylus, Typescript, UglifyJS, and YUI Compressor JS.
Show console	Use this drop-down list to define when you want the File Watcher open its dedicated console with messages. The available options are: – Always: when this option is chosen, the File Watcher opens the console when it starts. – Error: when this option is chosen, the File Watcher opens the console only if any errors occur in transpilation. – Never: choose this option to suppress opening the console under any circumstances.
Check file for syntax errors	– Select this check box to have the File Watcher ignore update, save, and change focus events in files that are syntactically invalid. – If this check box is cleared, the File Watcher starts anyway upon update, save, or frame deactivation, depending on the status of the Immediate file synchronization check box.

Watcher Settings Area

In this area, configure interaction with the transpiling tool: specify the executable file to use, the arguments to pass to it, and customize the default template settings for input and output.

Item	Description
File type	Use this drop-down list to specify the expected type of input files. The File Watcher will consider only files of this type as subject for analyzing and processing. File types are recognised based on associations between file types and file extensions . By default, the field shows the file type in accordance with the chosen predefined File Watcher template.
Scope	Use this drop-down list to define the range of files the File Watcher can be applied to. Changes in these files will invoke the File Watcher either immediately or upon save or frame deactivation, depending on the status of the Immediate file synchronization check box. You can choose one of the predefined scopes from the drop-down list: – Project Files: all the files within the project content roots (see Content Root and Configuring Content Roots). – Project Production Files: all the files within the project content roots excluding test sources. – Project Test Files: all the files within the project test source roots. – Open Files: all the files that are currently opened in the editor.

VCS Scopes: these scopes are only available if your project is under version control .

- Changed Files: all changed files, that is, all files associated with all existing changelists .
- Default: all the files associated with the changelist `Default` .

Alternatively, click the Browse button and configure a custom scope in the Scopes dialog box that opens.

For more details on scopes, see the pages Scopes and Scopes dialog .

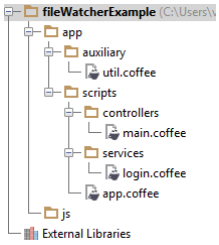
The Scope setting overrides the Track only root files check box setting: if a dependency is outside the specified scope, the File Watcher is not applied to it.

Program	In this text box, specify the location of the transpiler's executable file (`.exe` , `.cmd` , `.bat` , or other depending on the specific tool. `.jar` archives are also acceptable but defining `PATH` variables for them is not supported.) Do one of the following: – Type the path explicitly. – Click the Browse button ... to open the Select Path dialog box and navigate to the desired location. – Click the Insert Macro button to open the Macros dialog box where you can select the relevant macro from the list.
Arguments	In this text box, define the arguments to pass to the transpiler and thus influence its behaviour. Among other cases, use this text box to change the default output location, that is, specify a custom location where you want the transpiler to store the files generated during transpilation. Do one of the following: – Type the list of arguments in the text box. – Click the Insert Macro button to open the Macros dialog box where you can select the desired macro from the list. When specifying the arguments, follow these rules: – Use spaces to separate individual arguments. – If an argument includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"` . – If an argument includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value"\` .
Working directory	In this text box, specify the directory to which the transpiler will be applied. Because the tool is always invoked in the context of a file, the default working directory is the directory of the current file. This setting is specified in all predefined templates through a macros `$FileDir$` . To update this default settings, do one of the following: – Type the path explicitly in the text box. – Click the Browse button ... to open the Select Path dialog box and navigate to the desired location. – Click the Insert Macro button to open the Macros dialog box, where you can select the desired macro from the list. Tip If the field is left blank, PhpStorm uses the directory of the file where the File Watcher is invoked.
Environment variables	Use this text box to specify a the `PATH` variable for a tool that is required for starting the transpiler but is not referenced in the path to it.
Output paths to refresh	In this text box, specify the files where the transpiler stores its output: the resulting source code, source maps, and dependencies. In other words, tell PhpStorm where it should search for the files generated through transpilation. Please note, that changing the value in this text box does not make the transpiler store its output in another location. To do that, specify the desired output location in the Arguments text box. Do one of the following: – Type the output paths manually. Use colons as separators. – Click the Insert Macro button to open the Macros dialog box, where you can select the desired pattern from the list.
Create output from stdout	– When this check box is selected, PhpStorm reads the native transpiler's output (`standard output stream (stdout)`) and generates the resulting files from it. – When the check box is cleared, the transpiler writes its output directly to the files specified in the Output paths to refresh field. Tip Some transpilers generate a `standard output stream (stdout)` file, others do not, which may lead to errors. Therefore it is strongly recommended to preserve the default setting.

Examples of customizing the behaviour of a transpiler

Any transpiler is an external, third-party tool. Therefore the only way to influence a transpiler is pass arguments to it just as if you were working in the command line mode. These arguments are specific for each tool. Below are two examples of customizing the default output location for the CoffeeScript transpiler.

Suppose, you have a project with the following folder structure:



By default, the generated files will be stored in the folder where the original file is. You can change this default location and have the generated files stored in the `js` folder. Moreover, you can have them stored in a flat list or arranged in the folder structure that repeats the original structure under the `app` node.

– To have all the generated files stored in the output `js` folder without retaining the original folder structure under the `app` folder:

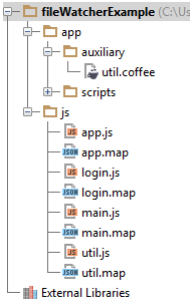
1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\ --compile --map $FileName$
```

2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileNameWithoutExtension$.map
```

As a result, the project tree looks as follows:



- To have the original folder structure under the `app` node retained in the output `js` folder:

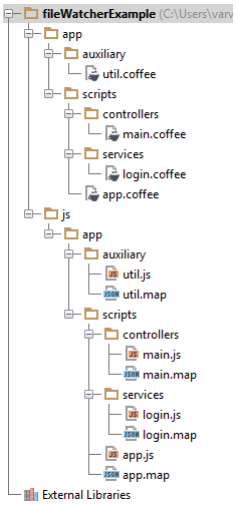
1. In the Arguments text box, type:

```
--output $ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\ --compile --map $FileName$
```

2. In the Output paths to refresh text box, type:

```
$ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileDirRelativeToI
```

As a result, the project tree looks as follows:



External Tools

File | Settings | Tools | External Tools for Windows and Linux

PhpStorm | Preferences | Tools | External Tools for OS X

`Ctrl+Alt+S`

Define third-party standalone applications (code generators and analyzers, pre- and post- processors, database utilities, etc.) as external tools to be able to run them from PhpStorm.

You can pass contextual information (like the currently selected file, or your project source path) to the external tools, view the tool output, and more.

The tools defined on this page appear as commands in the Tools menu and in various context menus. They can also be assigned keyboard shortcuts (see Configuring Keyboard Shortcuts).

Toolbar icons

Icon	Description
+	Add a definition of an external tool. (The Create Tool dialog will open.)
−	If an individual tool is selected: delete the tool definition. If a tool group is selected: delete the definitions of all the tools within the selected group.
	Edit the definition of the selected tool. (The Edit Tool dialog will open.)
	Move the selected tool one line up or down within the group. (The order of tools defines the order of items in corresponding menus.)
	Create a copy of the selected definition and then edit that copy. (The Copy Tool dialog will open.)

Check boxes

Use the check boxes to enable or disable the tools and the tool groups. The items that are not currently selected are not available in the Tools and context menus.

Create/Edit/Copy Tool Dialog

From the External Tools page : , or

Edit the settings for your external tool.

Item	Description
Name	The name of the tool that appears as a command name in the Tools menu and the context menus. See also, Show in .
Group	The group the tool belongs to. The tool groups correspond to submenus in the Tools menu and the context menus. Select an existing group from the list or type the name for a new group.
Description	The tool description (optional).
Options	
Synchronize files after execution	Make PhpStorm aware of changes in the file system when the tool completes its execution.
Open console	Open the console for viewing the tool output such error messages, etc.
Output Filters	Open the Output Filters dialog to manage the output filters associated with the tool. (The output filters are used to turn absolute file paths and line numbers in the tool output into hyperlinks. You'll be able to use those links to open the corresponding files in the editor.)
Show console when a message is printed to standard output stream	Make the output console active and bring it forward when the corresponding event occurs.
Show console when a message is printed to standard error stream	The same as the previous option but for stderr.
Show in	Specify in which menus the command for running the tool should be included. Main menu means the Tools menu. The rest of the options correspond to context menus in various places.
Tool settings	
Program	The path to the executable file to be run. Use ... to select the file, Insert macro to open the Macros dialog to select a macro. (Macros are resolved at runtime and let you specify context information such as currently selected file, your project source paths, etc.)
Parameters	The parameters to be passed to the program the way you'd specify them on the command line. Use Insert macro to open the Macros dialog to select a macro. When specifying the parameters, follow these rules: – Use spaces to separate individual parameters. – If a parameter includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"` . – If a parameter includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value\"` .
Working directory	The path to the current working directory for the program. Use ... to select the directory, Insert macro to open the Macros dialog to select a macro.

Output Filters Dialog

From the Create/Edit/Copy Tool dialog : Output Filters

This dialog lets you manage the output filters associated with an external tool. (The output filters are used to turn absolute file paths and line numbers in the tool output into hyperlinks.)

Item	Description
+	Create a new filter. (The Add Filter dialog will open.)
−	Delete the selected filter.
	Edit the selected filter. (The Edit Filter dialog will open.)
	Move the selected filter one line up or down in the list. (For each line in the output, the first matching filter is used.)

Add/Edit Filter Dialog

From the Output Filters dialog : + or

Specify the filter for transferring absolute file paths and line numbers in the tool output into hyperlinks.

Item	Description
Name	The name of the filter.
Description	The filter description (optional).
Regular expression to match output	The text pattern to be matched against the tool output to identify linkable references. The pattern can include text and the following placeholder variables: – `$FILE_PATH$` - the portion of the output that corresponds to an absolute path to a source file. Required. – `$LINE$` - a line number reference. – `$COLUMN$` - a column reference. The variables are inserted by right-clicking the field and then selecting the necessary item from the list that is shown. Example If your tool outputs the lines similar to `Error parsing C:\Demos\src\converter\MetersToInches.xml:103 Missing Closing Tag` the pattern `$FILE_PATH$:$LINE$` will turn `C:\Demos\src\converter\MetersToInches.xml:103` into a hyperlink to the line number `103` in the file `MetersToInches.xml`.

Macros Dialog

From the Create/Edit/Copy Tool dialog : Insert macro

Select the macro to be inserted.

Item	Description
Macros	The list of available macros with their descriptions.
Macro preview	The value of the selected macro in the current context.

Terminal

File | Settings | Tools | Terminal for Windows and Linux

PhpStorm | Preferences | Tools | Terminal for OS X

`Ctrl+Alt+S`

Use this dialog to specify settings for the embedded local terminal.

Prerequisites

Before you start working with terminal, make sure that the Terminal plugin is enabled. The plugin is bundled with PhpStorm and is activated by default. If it is not, enable the plugin as described in Enabling and Disabling Plugins .

Item	Description
Shell path	Specify the shell that will run by default.
Tab name	Specify the default name of a new session tab. Note that a session tab can be renamed.
Close session when it ends	If this check box is selected, the current session ends automatically, when the corresponding process ends (for example, by kill).
Audible bell	If this check box is selected, the console plays the bell sound on incoming escape sequence.
Mouse reporting	If this check box is selected, the embedded local terminal supports mouse pointer.
Copy to clipboard on selection	If this check box is selected, the text selected in the Terminal, is automatically passed to the system clipboard.
Paste on middle mouse button click	If this check box is selected, you can paste the clipboard contents by clicking the middle mouse button.
Override IDE shortcuts	If this check box is selected (this is the default setting), the Terminal tool window handles keyboard shortcuts differently from PhpStorm. If this check box is cleared, the PhpStorm key bindings are used.

Database

File | Settings | Tools | Database for Windows and Linux

PhpStorm | Preferences | Tools | Database for OS X

From the database console :

From the table editor : | Settings

The Database and subordinate pages contain the settings related to working with databases and SQL.

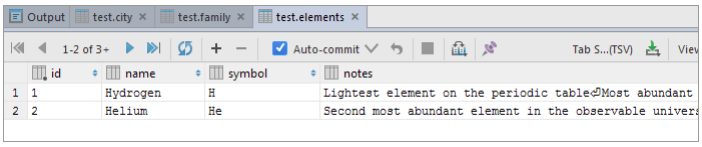
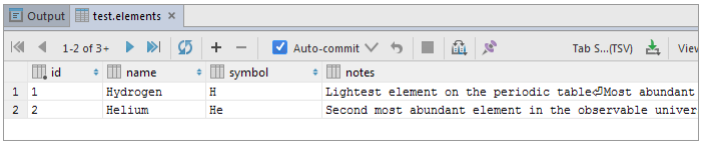
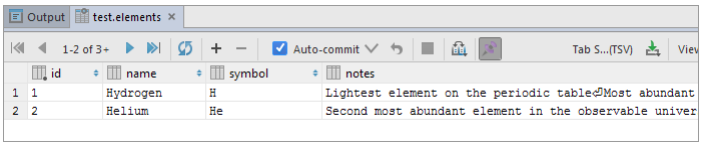
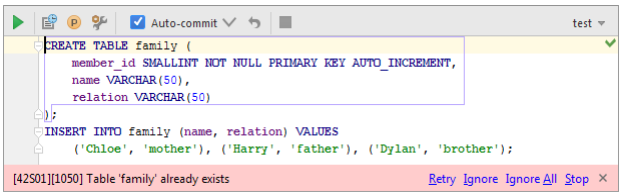
- Console
- Execute in Console
- Quick Documentation

- Data Views
- User Parameters
- CSV Formats

See also, SQL Dialects .

Console

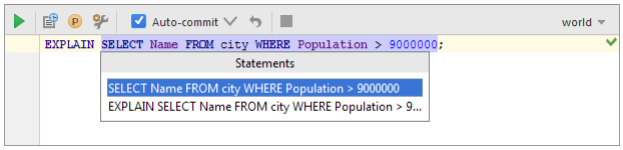
The settings in this section relate to showing various information in database consoles .

Item	Description
Show query results in new tab	You can select to view query results on individual tabs, or on one and the same tab. If the check box is selected, a new tab with the query result will open each time you run a query (`SELECT`). In this way, you can keep the results of all the queries that you have run.  If this check box is not selected, the same tab is used to show your query results. When you run a query, the information on the tab is updated to show the result, and a new tab doesn't open.  In this case, when you get the result that you want to keep, you can pin the tab to the tool window (on the toolbar or Pin Tab in the context menu for the tab). 
Show error notifications in editor	If this check box is not selected, the information about the errors is shown only in the output pane. If, in addition, you want the error notification bar to appear in the input pane, select the check box. The error notification bar may be particularly useful when running sequences of SQL statements. If an error occurs in such cases, the error notification bar lets you select how to react.  For more information, see Using the error notification bar.
Always review parameters before execution	When you run a statement with parameters, PhpStorm memorizes the parameter values. Each next time you execute the statement: – If this option is on, PhpStorm shows you the last used parameter values so that you can change them before actually running the statement. – If this option is off, PhpStorm executes the statement right away without showing you the parameter values.

Execute in Console

This section contains the options for the Execute command.

Item	Description
When inside a statement execute	If the cursor is inside a statement, the following options are available: – Ask what to execute. A list of statements that you can run is shown and you can select the statement or statements.



- Smallest statement. The smallest of the possible statements is executed. For example, when the cursor is inside a subquery, the subquery is executed.
- Largest statement. The largest possible statement is executed. For example, when the cursor is inside a subquery, an outer statement is executed.
- Largest statement or batch. For Transact-SQL (SQL Server and Sybase), the current batch of statements is executed. For all other dialects - the same as the previous option.
- Whole script. All the statements are executed.

otherwise execute	If the cursor is outside of a statement, e.g. on a blank line or within a comment, the following options are available: – Nothing. None of the statements is executed. – Whole script. All the statements are executed. – Everything below caret. All the statements after the cursor position are executed.
for selection execute	If something is currently selected, the following options are available: – Exactly as one statement. Exactly what is selected is executed as a single statement. – Exactly as statements. Exactly what is selected is executed. If the selection contains more than one statement, the statements are executed as separate statements. – Smart expand to script. If there is at least one statement border within the selection, the selection is expanded to form a sequence of valid statements. This sequence is then executed. Otherwise, precisely what is selected is executed.

Quick Documentation

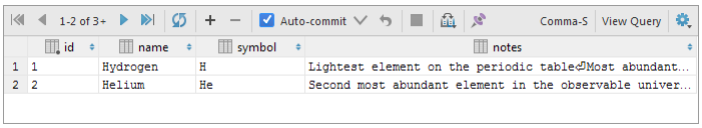
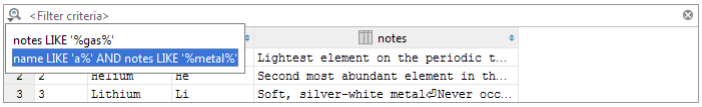
Item	Description
Allow database access	Let PhpStorm access the database when generating quick documentation. For example, for a table: – If this option is on, first rows are shown, and the table definition (`CREATE TABLE`) is retrieved from the database. – Otherwise, the data are not shown, and an auto-generated table definition is presented. Generally, when the option is on, you get more info but it takes more time to get the info.
Number of preview rows	The number of rows to be shown for a table in the quick documentation view.

Data Views

File | Settings | Database | Data Views for Windows and Linux

PhpStorm | Preferences | Database | Data Views for OS X

The settings on this page define how table data are shown in the database console , table editor and quick documentation preview.

ItemDescription	
Result set page size	The number of table rows to be shown at a time, on one "page". Here is an example when this number is set to 2:  If you don't want to limit the number of rows displayed simultaneously, specify zero (0).
Result set prefetch size	Data from databases are retrieved in chunks. The number in this field defines the number of rows in such chunks. A bigger number means fewer IDE - DB round trips but more memory for storing a chunk.
Filter history size	The number of most recently used filtering conditions to memorize for a table in the table editor. Here is an example when this number is set to 2. (The filter history box contains two most recently used conditions.) 
LOB length limit	The maximum size of a binary large object to be loaded in bytes.

User Parameters

File | Settings | Tools | Database | User Parameters for Windows and Linux

PhpStorm | Preferences | Tools | Database | User Parameters for OS X

Generally, only the question mark (`?`) is treated as a parameter in SQL statements. On this page, you can specify which other characters and their sequences should be treated as parameters, and in which places.

The patterns for SQL parameters are specified by means of regular expressions.

Item	Description
Enable in console and SQL files	If the check box is selected, the parameter patterns are applied to SQL (in SQL files and database consoles). The usage scope, if necessary, may be limited at the level of individual patterns . If this check box is not selected, the patterns are not used in SOL files and consoles irrespective of which usage scope is specified for individual patterns.
Enable in string literals with SQL injection	If the check box is selected, the parameter patterns are applied to string literals injected with SQL in the languages other than SQL. (See Using Language Injections .) The usage scope, if necessary, may be limited at the level of individual patterns . If this check box is not selected, the patterns are not used in string literals irrespective of which usage scope is specified for individual patterns.
Parameter patterns	The table shows the parameter patterns and their usage scopes. The patterns are specified using regular expressions. Values in parentheses are treated as parameter names. The patterns available initially have the following meanings: – `\?(\d+)` - a question mark followed by one or more digits, e.g. `?69` in which case `69` would be the parameter name. – `:(\w+)` - a colon followed by one or more word characters, e.g. `:x` , `:value` , `:parameter_1` . – `%w+` - `%` followed by one or more word characters, e.g. `%xyz` . To have PhpStorm recognize sprintf format references in injected SQL strings: 1. Click the `%w+` pattern. PhpStorm expands a hidden area with the configuration settings for the pattern. 2. Select the In scripts and In literals check boxes. 3. Click the <Language> link and select PHP in the drop-down list. – `\$\{([^$\{\}]*)\}` - `$` , then `{` , then any character except `$` , `{` or `}` zero or more times, then `}` , e.g. `${}` , `${value}` . – `\$\(([^\)]+)\)` - `$` , then `(` , then any character except `)` one or more times, then `)` , e.g. $(x). – `\$(\w+)\$` - `$` , then one or more word characters, then `$` again, e.g. `$x1$` . – `\#(\w+)\#` - `#` , then one or more word characters, then `#` again, e.g. `#field_3#` . Use + (Alt+Insert), — (Alt+Delete), ↑ (Alt+Up) and ↓ (Alt+Down) to add, delete and reorder the patterns. To edit a pattern or its usage scope, click the pattern and use the following controls: – In scripts. Clear this check box if the pattern shouldn't be used in SQL files and database consoles. – In literals. Clear this check box if the pattern shouldn't be used in string literals injected with SQL. – All (the link text may be different). Click the link and deselect the languages in which the pattern shouldn't be used.

CSV Formats

File | Settings | Tools | Database | CSV Formats for Windows and Linux

PhpStorm | Preferences | Tools | Database | CSV Formats for OS X

This page contains the settings for converting table data into delimiter-separated values formats (e.g. CSV, TSV) and vice versa.

When working on the conversion settings, use the preview in the right-hand part of the page.

Item	Description
Formats	The list of the available delimiter-separated values formats is shown. Each format is a named set of corresponding conversion settings. Select the format whose settings you want to view or edit. Use +, —, and to create, delete and reorder the formats; to create a copy of the selected format.
Value separator	Select or type the character for separating individual values.
Row separator	Select or type the character for separating rows.
Null value text	The text to be used as a value if a cell contains `null` (an unknown value).
Add row prefix/suffix	Row prefix and suffix are character sequences which in addition to the row separator indicate the beginning and end of a row. If necessary, click the link and specify the row prefix and suffix in the fields that appear.
Quotation	Each line in the area under Quotation is a quotation pattern (see Quote values). A quotation pattern includes: − The left quotation character, the one inserted before a value. − The right quotation character, the one inserted after a value; usually, the same as the left quotation character. − An escape method or character for the cases when the quotation character is part of a value. E.g. Escape: duplicate means that if a quotation character occurs within a value, it is doubled. (You can specify your own escape character instead.) If there is more than one pattern, the first of the patterns is used. Use +, —, and to create, delete and reorder the patterns. To start editing an existing pattern, just click the pattern of interest.
Quote values	Specify in which cases the values should be quoted (i.e. enclosed within quotation characters). − When needed. A value is quoted only if it contains the value and/or the row separator. − Always. Any value is quoted in its text representation.
Trim whitespaces	If this check box is not selected, the Unicode whitespace characters that precede and follow the value separators are treated as parts of the corresponding values. If this check box is selected, the corresponding whitespace characters are ignored or removed.
First row is header	If this check box is selected, the first row is treated as containing column names. The settings that appear under Header Format have the same meanings as the ones above but are applied to the first row.
First column is header	If this check box is selected, the first column is treated as containing row names.

SSH Terminal

File | Settings | Tools | SSH Terminal for Windows and Linux

PhpStorm | Preferences | Tools | SSH Terminal for OS X

Ctrl+Alt+S

Use this dialog box to appoint the a remote Web server or a Vagrant instance (virtual machine) to access through the SSH terminal, configure connection with the destination environment, and choose the encoding to use in the SSH terminal.

Make sure the SSH Remote Run plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Item	Description
Connection settings	In this section, appoint the a remote Web server or a Vagrant instance (virtual machine) to access through the SSH terminal and specify where the connection settings should be taken from: 1. Current Vagrant: select this option to have the commands in the SSH Terminal executed on the currently running Vagrant virtual machine. For details, see Vagrant: Working with Reproducible Development Environments . 2. Deployment server: select this option to have the commands in the SSH Terminal executed on the local or remote Web server accessible through one of the server access configurations . From the drop-down list, choose the server access configuration that specifies the destination environment and the settings to establish connection to it. – Select server on every run: if this option is selected, you will have to choose the desired server access configuration from the pop-up window, every time you choose Tools | Start SSH Session on the main menu. – If the desired server access configuration does not appear in the drop-down list, click the link Configure Servers, and define one in the Deployment page. For details, see Configuring Synchronization with a Web Server .
Default encoding	From this drop-down list, select the desired encoding to be used in the SSH terminal.

Command Line Tool Support

File | Settings | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Command Line Tool Support for OS X

Ctrl+Alt+S

This page shows a list of all PHP-specific and custom command line tools that are currently available at the PhpStorm level.

Tools with inconsistencies in the definition files are marked with the Invalid description icon.

Note that PHP-specific command line tools work only with local PHP interpreters, see Configuring Local PHP Interpreters and Enabling PHP Support .

Use this page to enable integration with third-party PHP command line tools and define your own ones.

Item	Tooltip and shortcut	Description
Table of available tools		
Enabled		When the check box in this column is selected, the commands defined within the corresponding command line tool can be executed from PhpStorm.
Alias		In a text box in this column, specify the character string to use in command calls instead of the full path to the tool. For example, by default, PhpStorm assigns the following aliases: – `s` for Symfony. – `zf` for Zend Framework. – `c` for Composer.
Tool Path		In a text box in this column, specify the location of the tool's executable file.
Type		A read-only field in this column shows the official name of the third party command line tool. 1. The column is available only if the Show tool type check box is selected. 2. For custom command line tools, the fields in this column are empty.
Toolbar		
+	Add Alt+Insert	Click this button to open the Command Line Tools dialog box and select the type of command-line tool to enable integration with. The available options are: – Custom Tool – Zend Framework 1 Tool – Zend Framework 2 Tool – Symfony – Composer – Drush – Tool based on Symfony Console Depending on your choice, PhpStorm opens one of the following dialog boxes for specifying the location of the selected tool: – Command Line Tool Support: Symfony – Command Line Tool Support: Zend Framework 1 – Command Line Tool Support: Composer – Command Line Tool Support: Tool Settings – Command Line Tool Support: Drush – Command Line Tool Support: Zend Framework 2 – Command Line Tool Support: WP-CLI
–	Remove Alt+Delete	Click this button to have the selected tool removed from the list.
	Edit	Click this button to change the definition of the selected tool.
	Reload command list from executable	Click this button to have PhpStorm refresh the list of commands of the selected tool.
	Open definition in editor	Click this button to have PhpStorm open the `.xml` file with commands of the selected tool in the editor.
Show tool type		If this check box is selected, the Type column is added to the table of available tools.
Show console in		In this area, specify where you want to enter commands. The available options are: – Pop-up - choose this option to have the Command Line Tools Input pane opened in a separate pop-up window and type commands there. – Tool window - choose this option to enter commands in the Input field in the bottom of the dedicated Command Line Tools Console tool window.
Console encoding		From this drop-down list, choose the character set to show the tool's output in the Command Line Tools Console Tool Window .

Command Line Tool Support: Symfony

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

The dialog box opens when you click the Add button and choose Symfony in the Choose Tool to Add dialog box.

Use the dialog box to configure Symfony support in PhpStorm.

Item	Description
Path to Symfony	In this text box, specify the location of the Symfony executable file: – `<symfony_home>/data/bin/symfony` for Symfony 1.2. – `<symfony_home>/data/bin/symfony.bar` for Symfony 1.4. – `<symfony_home>/app/console` for Symfony 2. Type the path manually or click the Browse button and choose the desired location in the dialog that opens . PhpStorm parses the contents of the specified file for Symfony commands.

Command Line Tool Support: Zend Framework 1

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

The dialog box opens when you click the Add button and choose Zend Framework 1 Tool in the Choose Tool to Add dialog box.

Use the dialog box to configure Zend Framework support in PhpStorm.

Item	Description
Path to *zf* tool	In this text box, specify the location of the `<Zend-Framework-home>/zf.bat` file. Type the path manually or click the Browse button ... and choose the desired location in the dialog that opens . PhpStorm parses the contents of the specified file for Zend Framework commands.

Command Line Tool Support: Composer

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Add button and choose Composer in the Choose Tool to Add dialog box.

In this dialog box, enable the use of Composer Dependency Manager in the command line mode by specifying the way to launch Composer and appointing the file to look for Composer commands in.

PhpStorm parses the contents of the specified `.phar` archive or executable file for Composer commands. When the file analyses is completed, PhpStorm returns to the Command Line Tools page of the Settings dialog bos where the specified file is added to the list of command line tools available in PhpStorm.

Integration with Composer is provided at the PhpStorm level, so once configured, the tool can be used in all your PhpStorm projects. Just activate or de-activate it when necessary depending on you needs.

Item	Description
Composer.phar or PHP script	Choose this option to launch Composer through a PHP script or have PhpStorm detect and start the launcher in the `composer.phar` archive. In this mode, PhpStorm provides coding assistance and allows to execute scripts.
Path to PHP executable	In this field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.
Path to composer.phar or composer	In this text box, specify the location of the `composer.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
Composer executable	Choose this option to launch Composer through the `Composer` executable file. In this mode, you do not get coding assistance and cannot execute scripts because not PHP engine is appointed for it.
Path to executable	In this text box, specify the location of the `composer` executable file. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.

Command Line Tool Support: Tool Settings

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

The dialog box opens when you click the Add button and choose Custom Tool in the Choose Tool to Add dialog box.

Use the dialog box to create your own command line tool.

Item	Description
Tool Name	In this text box, specify the name of the new tool.
Tool path	In this text box, specify the location of the new tool.
Alias	In this text box, type the character string to use in command calls instead of the full path to the tool.
Description	In this text box, provide a brief explanation of the tool functionality.

Command Line Tool Support: Drush

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

`Ctrl+Alt+S`

The dialog box opens when you click the Add button and choose Drush in the Choose Tool to Add dialog box. Use the dialog box to configure Drush support in PhpStorm.

Drush is a command line shell and scripting interface for Drupal . With PhpStorm, you can use Drush 5.8 and higher. For more details, see Using Drupal with PhpStorm and Drupal-Specific Coding Assistance .

Item	Description
Path to Drush	In this text box, specify the path to the Drush executable file. PhpStorm automatically fills in the default location, which is usually `C:/ProgramData/Drush/drush.bat` on Windows and `/usr/bin/drush` on Mac OS or Linux. If you followed the standard installation procedure, the predefined path will be correct, just click OK, whereupon PhpStorm loads command definitions automatically and returns to the Command Line Tool Support page.

Command Line Tool Support: Zend Framework 2

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

The dialog box opens when you click the Add button and choose Zend Framework 2 Tool in the Choose Tool to Add dialog box.

Use the dialog box to configure Zend Framework support in PhpStorm.

Item	Description
Path to *zf* tool	In this text box, specify the location of the `<Zend-Framework-home>/zf.php` or `<Zend-Framework-home>/zf.php` file. Type the path manually or click the Browse button and choose the desired location in the dialog that opens . PhpStorm parses the contents of the specified file for Zend Framework commands.

Command Line Tool Support: WP-CLI

File | Settings | Tools | Command Line Tool Support for Windows and Linux

PhpStorm | Preferences | Tools | Command Line Tool Support for OS X

The dialog box opens when you click the Add button and choose Composer in the Choose Tool to Add dialog box.

In this dialog box, enable the use of WordPress Content Management System in the command line mode by specifying the way to launch WordPress.

Item	Description
Installed via PHAR	Choose this option to launch WordPress through a PHP script or have PhpStorm detect and start the launcher in the `wp-cli.phar` archive.
Path to PHP executable	In this field, specify the location of the PHP engine installation folder. Choose the PHP installation from the list, or type the path manually, or click the Browse button ... and choose the desired location in the dialog box that opens.
Path to phar	In this text box, specify the location of the `wp-cli.phar` archive. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.
Executable available (installed via Composer, etc.)	Choose this option to launch WordPress through an executable file which is available when you install WordPress using a package management tool, for example, Composer.
Path to wp.bat	In this field, specify the location of the `wp.bat` or `wp` executable file. If you used Composer, the default location is `\vendor\wp\cli\bin\wp` or `\vendor\wp\cli\bin\wp.bat`. Type the path manually or click the Browse button ... and choose the desired location in the dialog box that opens.

Diagrams

File | Settings | Tools | Diagrams for Windows and Linux

PhpStorm | Preferences | Tools | Diagrams for OS X

`Ctrl+Alt+S`

Use this page to configure the default visibility settings and layout for diagrams.

On this page:

– Content pane

– Controls

Content pane

Select the check boxes next to the elements to be shown on diagrams.

Item	Description
Show Difference	
Details	If this check box is selected, all the specified details of the elements will be shown in the UML class diagram for a revision. If this check box is not selected, only node elements will be included in diagram.
DB Diagrams	
Key columns	For the primary key columns to be shown when a diagram opens, select this check box. When viewing a diagram in the editor, use on the toolbar to show or hide the corresponding columns.
Columns	For the columns other than the primary key columns to be shown when a diagram opens, select this check box. When viewing a diagram in the editor, use on the toolbar to show or hide the corresponding columns.

Note More nodes appear in this pane depending on the installed and enabled plugins.

Controls

Item	Description
Default layout	Select the desired layout from the drop-down list. Node elements in newly created diagrams will be arranged according to the selected layout.
Default scope	Select scope from the drop-down list. Specifying a scope helps you avoid showing in diagram the unnecessary hierarchies. You can define scopes for your project in the Scopes page of the Settings dialog.
Fit content after layout	If this check box is selected, then after applying a layout selected on the diagram context menu, all diagram elements will be resized to fit into the current diagram area. In diagram, use the toolbar button.
Do relayout when new elements were added	If this check box is selected, diagram layout will be performed automatically after adding new elements.
Enable colors	If this check box is selected, relationship links will be shown colored.

Diff & Merge

File | Settings | Tools | Diff & Merge for Windows and Linux

PhpStorm | Preferences | Tools | Diff & Merge for OS X

`Ctrl+Alt+S`

On this page, specify the default behavior of the Differences viewer .

Item	Description
Diff	
Context lines	Use the slider to specify the amount of context lines that will not collapse, when in a Differences viewer you click the button to collapse the unchanged fragments .
Go to the next file after reaching last change	If this check box is selected, PhpStorm will suggest to click `F7` / `Shift+F7` once more and compare other files
Merge	
Automatically apply non-conflicting changes	If this check box is selected, the interactive merge tool automatically merges all non-conflicting changes. This is an equivalent to clicking in the Merge dialog.

External Diff Tools

File | Settings | Tools | Diff | External Diff Tools for Windows and Linux

PhpStorm | Preferences | Tools | Diff | External Diff Tools for OS X

`Ctrl+Alt+S`

If necessary, specify external tools for comparing files and folders, and associated settings.

Item	Description
Use external diff tool	Select this check box to have PhpStorm use an external tool for comparing files or folders.
Path to executable	In this text box, specify the path to the executable file of the desired external diff tool. Use the Browse button ..., if necessary. This field is only available, when Use external diff tool check box is selected.
Use by default	Select this check box to have PhpStorm use the specified external tool for comparing files or folders by default. If this check box is not selected, PhpStorm will use the built-in diff tool . To invoke an external tool, click the button on the toolbar of the Differences viewer . This field is only available, when Use external diff tool check box is selected.
Parameters	Use this field to set the diff tool parameters. **Note** Note that different diff tools have different parameters. You need to specify all the necessary parameters in proper order. This field is only available, when Use external diff tool check box is selected.
Use external merge tool	Select this check box to have PhpStorm use an external merge tool. In the text box below, specify the path to the executable file of the desired external tool. Use the Browse button ..., if necessary.
Path to executable	In this text box, specify the path to the executable file of the desired external merge tool. Use the Browse button ..., if necessary. This field is only available, when Use external merge tool check box is selected.
Parameters	Use this field to set the merge tool parameters. **Note** Note that different merge tools have different parameters. You need to specify all the necessary parameters in proper order. This field is only available, when Use external merge tool check box is selected.

Docker Machine

File | Settings | Tools | Docker Machine for Windows and Linux

PhpStorm | Preferences | Tools | Docker Machine for OS X

For this page to be available, the Docker integration plugin must be installed.

Specify the settings for working with Docker Machine .

Item	Description
Docker Machine executable	`docker-machine` or an actual path to `docker-machine.exe` (normally located in the Docker Toolbox installation folder). The default setting `docker-machine` is fine if: – The actual name of the executable file is `docker-machine` . – The path to the directory where the file is located is included in the environment variable `Path` .
Detect	Set Docker Machine executable to `docker-machine` , try to detect the executable file and, if a success, show its version.

Remote SSH External Tools

File | Settings | Tools | Remote SSH External Tools for Windows and Linux

PhpStorm | Preferences | Tools | Remote SSH External Tools for OS X

Define remote applications that require SSH access as external tools to be able to run them from PhpStorm.

You can pass contextual information (like the currently selected file, or your project source path) to the external tools, view the tool output, and more.

The tools defined on this page appear as commands in the Tools menu and in various context menus. They can also be assigned keyboard shortcuts (see Configuring Keyboard Shortcuts).

Toolbar icons

Icon	Description
+	Add a definition of an external tool. (The Create Tool dialog will open.)
−	If an individual tool is selected: delete the tool definition. If a tool group is selected: delete the definitions of all the tools within the selected group.
	Edit the definition of the selected tool. (The Edit Tool dialog will open.)
	Move the selected tool one line up or down within the group. (The order of tools defines the order of items in corresponding menus.)
	Create a copy of the selected definition and then edit that copy. (The Copy Tool dialog will open.)

Check boxes

Use the check boxes to enable or disable the tools and the tool groups. The items that are not currently selected are not available in the Tools and context menus.

Create/Edit/Copy Tool Dialog (Remote SSH External Tools)

From the Remote SSH External Tools page : , or

Edit the settings for your external tool.

Item	Description
Name	The name of the tool that appears as a command name in the Tools menu and the context menus. See also, Show in .
Group	The group the tool belongs to. The tool groups correspond to submenus in the Tools menu and the context menus. Select an existing group from the list or type the name for a new group.
Description	The tool description (optional).
Options	
Synchronize files after execution	Make PhpStorm aware of changes in the file system when the tool completes its execution.
Open console	Open the console for viewing the tool output such error messages, etc.
Output Filters	Open the Output Filters dialog to manage the output filters associated with the tool. (The output filters are used to turn absolute file paths and line numbers in the tool output into hyperlinks. You'll be able to use those links to open the corresponding files in the editor.)
Show console when a message is printed to standard output stream	Make the output console active and bring it forward when the corresponding event occurs.
Show console when a message is printed to standard error stream	The same as the previous option but for stderr.
Show in	Specify in which menus the command for running the tool should be included. Main menu means the Tools menu. The rest of the options correspond to context menus in various places.
Connection settings	In this section, appoint the to access through the SSH terminal and specify where the connection settings should be taken from: 1. Current Vagrant: select this option to have the commands in the SSH Terminal executed on the currently running Vagrant virtual machine. For details, see Vagrant: Working with Reproducible Development Environments . 2. Deployment server: select this option to have the commands in the SSH Terminal executed on the local or remote Web server accessible through one of the server access configurations . From the drop-down list, choose the server access configuration that specifies the destination environment and the settings to establish connection to it. – Select server on every run: if this option is selected, you will have to choose the desired server access configuration from the pop-up window, every time you choose Tools | Start SSH Session on the main menu. – If the desired server access configuration does not appear in the drop-down list, click the link Configure Servers, and define one in the Deployment page. For details, see Configuring Synchronization with a Web Server .
Tool settings	
Program	The path to the executable file to be run. Use ... to select the file, Insert macro to open the Macros dialog to select a macro. (Macros are resolved at runtime and let you specify context information such as currently selected file, your project source paths, etc.)
Parameters	The parameters to be passed to the program the way you'd specify them on the command line. Use Insert macro to open the Macros dialog to select a macro. When specifying the parameters, follow these rules: – Use spaces to separate individual parameters. – If a parameter includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"` . – If a parameter includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value\"` .
Working directory	The path to the current working directory for the program. Use ... to select the directory, Insert macro to open the Macros dialog to select a macro.

Server Certificates

On this page:

- Access
- Overview
- Table of controls

Access

File | Settings | Tools | Server Certificates for Windows and Linux

PhpStorm | Preferences | Tools | Server Certificates for OS X

`Ctrl+Alt+S`

Overview

PhpStorm provides its own storage for trusted certificates. Use this page to manage this storage.

Table of controls

Item	Shortcut	Description
Accept non-trusted certificates automatically		Select this option if you want non-trusted certificates (i.e. the certificates that are not added to the list) to be accepted automatically, without sending a request to the server.
+	`Alt+Insert`	Add a trusted server certificate to the list. Select the certificate file in the dialog that opens . The certificate file should have an extension `.crt` , `.cer` or `.pem` . For a trusted certificate, the certificate information is shown in the lower part of the page.
−	`Alt+Delete`	Remove the selected trusted certificate from the list.

Settings Repository

File | Settings | Tools | Settings Repository for Windows and Linux

PhpStorm | Preferences | Tools | Settings Repository for OS X

Ctrl+Alt+S

This page appears in the Settings/Preferences dialog, when the Settings Repository plugin is enabled.

The plugin is bundled with PhpStorm and is activated by default. If it is disabled, you can manually enable the plugin .

Use this page to configure the Settings Repository feature that allows you to share your IDE settings between different instances of PhpStorm (or other IntelliJ platform-based) products installed on different computers.

Tip The settings you are going to share must be stored in a Git repository.

Item	Description
Auto Sync	Select this check box, if you want your local settings to be automatically synchronized with the settings stored in the repository every time you perform an Update Project or a Push operation, or when you close your project or exit PhpStorm. If this option is disabled, you can manually update your settings by choosing VCS | Sync Settings from the main menu.
Read-only sources	Use this section to configure additional repositories containing any types of settings you want to share, including live templates, file templates, schemes, deployment options, etc. These repositories cannot be overwritten or merged, just used as a source of settings as is. Use the following controls to manage the read-only repositories: + Click this button to add the URL of the GitHub repository that contains the settings you want to share. − Click this button to remove the selected repository from the list. Click this button to edit the URL of the selected source. / Use these buttons to move up/down in the list. Click this button to clone the selected URL.

Startup Tasks

On this page, create a list of run/debug configurations to be launched automatically on the project start. This may be helpful if you run some Grunt or Gulp.js tasks or npm scripts on a regular basis. All you need is just add the run/debug configurations that launch such tasks or scripts to the list of startup tasks.

On this page:

– Access

– Table of Controls

Access

File | Settings | Tools | Startup Tasks for Windows and Linux

PhpStorm | Preferences | Tools | Startup Tasks for OS X

`Ctrl+Alt+S`

Table of Controls

ItemTooltipDescription		
Run Configuration		This read-only field shows the names of the run configurations to be launched on the project start.
Shared		When this check box is selected, the corresponding task is available for other team members. Note that you can share only those run configurations that are already marked as shared in their definitions. The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml`.
+	Add	Click this button to add one of the run/debug configurations that are currently defined in the project to the list of tasks to be executed on the project start. Choose the relevant configurations from the list or choose Edit Configurations to open the Run/Debug Configurations dialog and define a required configuration in it.
–	Remove	Click this button to delete the selected task from the list.
	Edit	Click this button to open the Run/Debug Configurations dialog with the settings from the selected configuration and update them as required.

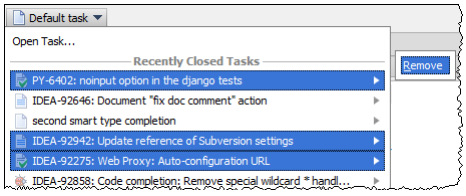
Tasks

File | Settings | Tools | Tasks for Windows and Linux

PhpStorm | Preferences | Tools | Tasks for OS X

`Ctrl+Alt+S`

Use this page to set up the general options for tasks and context management .

Item	Description
Changelist name format	Type here the format of a changelist to be created. The available placeholders are: `{id},{summary},{number}, {project}`
Task history length	Specify the number of tasks to be stored in history.
Connection timeout	Sets the maximum time for a connection to be established. A value of zero means the timeout is not used. The default setting is five seconds (5000).
Always display task combo in toolbar	If this check box is selected, the combo box of tasks appears in the main toolbar. Use this combo box to switch between tasks , and remove the unnecessary tasks:  If this check box is not selected, then the tasks combo box hides, when the only open task is the default one.
Save context on commit	If this check box is selected, a new task with the name of the committed changelist is created upon commit. The new task will be available in the tasks combo on the main toolbar.
Enable issue cache	– Select this check box to have PhpStorm synchronize with the issue tracking system in the background on a regular basis, No matter whether you actually request on information from your issue tracker or not, PhpStorm will connect to your issue tracking system according to the specified frequency and refresh the cached issues. The advantage of this approach is that when you need to switch to a task, the up-to-date information is already at your disposal so you do not need to wait till PhpStorm establishes connection with the tracker and retrieves the information. Tip This configuration is especially recommended when working with rather "slow" issue tracking systems. – Clear this check box to have PhpStorm connect to the issue tracking system only when you actually need information on issues, clear the Enable issue cache check box.
Update (N) issues each (M) minutes	Specify how often PhpStorm should synchronize with your issue tracking system and refresh the cached issues.

Servers

File | Settings | Tools | Tasks | Servers for Windows and Linux

PhpStorm | Preferences | Tools | Tasks | Servers for OS X

`Ctrl+Alt+S`

Use this page to associate your account in the bug tracking system you use with the PhpStorm project. The page is divided into the following sections:

- Configured servers
- Server options
  - General tab
  - Commit Message tab
  - Server Configuration tab

Configured servers

Item	Shortcut	Description
+	`Alt+Insert`	Click to add a new server of one of the supported types. For example: – YouTrack . Make sure that REST API in your YouTrack server is enabled. – JIRA – GitHub – Generic Click + to explore the list of the supported servers.
–	`Alt+Delete`	Click to remove the selected server.

Server options

General tab

Item	Available for	Description
Share URL	All servers	If this check box is selected, the server URL of your issue tracking system, specified in the field below becomes available to the other team members.
Server URL	All servers	Specify here the server URL of your issue tracking system. Tip Important note for YouTrack users: If you are using free instance of YouTrack InCloud, make sure to specify `http://<sitename>/youtrack` . If you are using a paid instance, specify `https://<sitename>/youtrack` .
Username / Password	All except for Lighthouse, Trello, GitLab, GitHub	Specify here login name and password for your account. Tip Important note for Sprintly users: The Password required here is the API Key from your Sprintly profile (not your Sprintly password). Refer to the Sprintly documentation for details.
Repository	GitHub	Specify here the name of the desired Git repository.
Search	YouTrack	Specify here the default search string.
Project ID	Lighthouse, Pivotal Tracker, Redmine	Specify here the name of your project in the tracker.
API Token	Lighthouse, Pivotal Tracker	Specify here the desired API token for the project in question.
Use proxy	All servers	Select this check box to use the specified proxy server.
Proxy settings	All servers	Click this button to set up proxy settings .
Use HTTP authentication	JIRA	Select this check box to enable HTTP authentication.
Test	All servers	Click this button to make sure the connection is established.
<Template variables>		More fields may be added to this tab depending upon the selected or cleared check box Show on first tab in the Template Variables dialog in the Server Configuration tab.

Commit Message tab

Item	Description
Add commit message	If this check box is selected, a commit message for a changelist, created from an issue in a bug tracking system, is formed according to the syntax defined below.
Message	Use the placeholders to define the commit message syntax. The available placeholders are `{id}` , `{summary}` , `{number}` , `{project}` .

Server Configuration tab

This tab only shows for the following servers:

- Generic
- Asana

- Assembla
- Sprint.ly

ItemDescription	
Login URL	Specify here the resource for authentication (if any). The request to this resource will take place every time the issues are retrieved from server. This field is disabled, if the check box Use HTTP authentication in the General tab is selected. The resource consists of a URL with possible template variables: – `{serverUrl}` : URL of the server, specified in the General tab. – `{password}, {username}` : correspond to the settings specified in the General tab. – `{max}` : maximum number of issues to be retrieved from the server. Also it contains the combo-box with the request type: GET or POST.
Tasks list URL	This resource enables obtaining the list of issues from the server and is requested every time the user opens tasks. The resource consists of a URL with possible template variables: – `{serverUrl}` : URL of the server, specified in the General tab. – `{password}, {username}` : correspond to the settings specified in the General tab. – `{max}` : maximum number of issues to be retrieved from the server. Also it contains the combo-box with the request type: GET or POST.
Single task URL	This resource makes it possible to obtain a more detailed information about a specific issue, using its ID. This field is mandatory, when the check box Each task in separate request is selected. The resource consists of a URL with possible template variables: – `{serverUrl}` : URL of the server, specified in the General tab. – `{password}, {username}` : correspond to the settings specified in the General tab. – `{max}` : maximum number of issues to be retrieved from the server. Also it contains the combo-box with the request type: GET or POST.
Response type	This section contains three radio buttons: – XML: click this radio button to return the server response in XML format. So doing, the selectors are described in the format XPath . – JSON: click this radio button to return the server response in JSON format. So doing, the selectors are described in the format JSONPath . – Text: click this radio button to retrieve issues fields using regular expressions.
Each task in separate request	Select this check box to make the field Single task URL mandatory.
Table of selectors	These selectors help specifying how exactly the detailed issue information will be extracted from the server response. The selectors have the following meanings: – tasks: path in the server response to the descriptions of specific issues (mandatory selector). – id: path to the unique ID within the description of a specific issue (mandatory selector). – summary: path to the title/header of an issue inside its description (mandatory selector). – description: path to the detailed description of an issue (optional selector). – updated/created: path to the time of updating or creating the issue respectively, in the format ISO-8601, inside its description (optional selector). – closed: path to a flag that defines whether a specific issue is opened or closed (optional selector). – issueUrl: path to a URL inside the issue description. It is possible to open information about this issue in your browser . Depending on the choice of response type , these selectors are described in XPath or JSONPath format, or as regular expressions.
Reset to Defaults	Click this button to discard all settings and return to the default values.
Manage Template Variables...	Click this button to show the dialog box Template Variables with the list of existing template variables. Use + and — buttons to manage the list. Note the check boxes in the columns Show on first tab and Hide. – If a check box in the column Show on first tab next to a custom template variable is selected, this custom variable is displayed in the General tab. – If a check box in the column Hide next to a custom template variable is selected, the value of this variable is shown dots. Otherwise, it is shown as is. Note that default variables cannot be deleted or changed!
Test	Click this button to make sure the settings are correct.

Time Tracking

File | Settings | Tools | Tasks | Time Tracking for Windows and Linux

PhpStorm | Preferences | Tools | Tasks | Time Tracking for OS X

`Ctrl+Alt+S`

Use this page to enable time tracking and configure its settings.

Item	Description
Enable Time Tracking	Select this check box to enable time tracking in the current project. When this check box is selected, the Time Tracking tool window becomes available on the right tool window bar, and on the View	Tool Windows menu.
Suspend delay	In this field, enter the "accuracy" of time tracking. If no user activity detected during this time, task is considered suspended.

Vagrant

File | Settings | Vagrant for Windows and Linux

PhpStorm | Preferences | Vagrant for OS X

Ctrl+Alt+S

Use this page to:

- Enable Vagrant support in PhpStorm by specifying the Vagrant executable file.
- Specify the location of the VagrantFile that determines creation of virtual boxes (instances) by provisioning predefined Vagrant base boxes through the `vagrant up` command.
- Handle the list of Vagrant base boxes to use in creation of virtual boxes (instances).

Before you start working with Vagrant, make sure that:

1. Vagrant is downloaded and installed.

Item	Description
Vagrant executable	Specify the fully qualified address of the executable file: `vagrant.bat` for Windows, `vagrant` for Unix and OS X. Type the path manually, or click the browse button and locate the desired file in the Select vagrant executable dialog box.
Instance folder	Specify here the fully qualified path to the directory, where the task `vagrant init` has been executed, and the `VagrantFile` is initialized and stored. A `VagrantFile` is a configuration file that defines the instance (virtual machine) you need. The file contains the virtual IP address, port mappings, and the memory size to assign. The file can specify which folders are shared and which third-party software should be installed. According to the `VagrantFile` your instance (virtual machine) is configured, provisioned against the relevant Vagrant base box, and deployed on your computer. A `VagrantFile` is created through the `vagrant init` command. When creation of an instance (virtual machine) is invoked either through the `vagrant up` command or through the Tools | Vagrant | Up menu option, PhpStorm looks for the `VagrantFile` in the directory specified in the Instance folder field. For more information, see http://docs.vagrantup.com/v2/vagrantfile/ . You can create a `VagrantFile` in any directory and appoint it as instance folder. If the field is empty, PhpStorm will treat the project root as the instance folder and look for a `VagrantFile` in it.
Provider	Use this field to specify the provider to be used by `vagrant up` command. If this field is left blank, the default provider is used.
Environment variable	Click the ellipsis button or press Shift+Enter to specify the shell variables to be used to configure the providers' behavior.
Boxes and Plugins tabs	
Boxes	This list shows the predefined Vagrant base boxes available in PhpStorm. Each item presents a Vagrant base box on which Vagrant configures and launches its instances (virtual machines). The entries of this list correspond to the output of the command `vagrant box list` . + Alt+Insert Click this button to download a new base box. This command corresponds to `vagrant box add <name> <URL>` . By default, PhpStorm suggests the URL to the lucid32 box − Alt+Delete Click this button to remove the selected Vagrant base box. So doing, the box and the nested files are physically deleted from the disk. This command corresponds to `vagrant box remove <name>`
Plugins	Use this table to view and change the list of available plugins. + Alt+Insert Click this button to install a new Vagrant plugin. − Alt+Delete Click this button to remove the selected plugin. Click this button to update the selected plugin. Use this button to attach a license to the selected plugin.

WordPress

File | Settings | Languages and Frameworks | WordPress for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | WordPress for OS X

`Ctrl+Alt+S`

The page is available only when the WordPress Support plugin is enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to enable the use of the WordPress Content Management System through the PhpStorm user interface. To use WordPress in the command line mode, configure it as a command line tool in Command Line Tool Support: WP-CLI .

Item	
Enable WordPress Integration	Select this check box to enable the use of the specified WordPress installation through the PhpStorm interface.
WordPress Installation Path	In this text box, specify the folder where WordPress is installed. This folder should contain the `wp-admin` and `wp-includes` subdirectories.

XPath Viewer

File | Settings | Tools | XPath Viewer for Windows and Linux

PhpStorm | Preferences | Tools | XPath Viewer for OS X

Ctrl+Alt+S

This page appears if XPath View+XSLT Support plugin is enabled. In this page, configure the PhpStorm behaviour during interactive execution of XPath expressions.

Item	Description
Scroll first hit into visible area	Select this check box to have the editor automatically scroll to the first XPath match.
Use node at cursor as context node	Select this check box to have the entered XPath expression use the currently selected node (tag/attribute/pi, etc.) as its context node and evaluate the expression relatively to this node.
Highlight only start tag instead of whole tag content	Do one of the following: – Select this check box to have only the name of a matching tag highlighted. – Clear this check box to have the entire content of a matching tag highlighted.
Add error stripe markers for each result	Select this check box to have each match supplied with an error stripe marker which can be quickly navigated to. The tooltip of each marker shows the matched content.
Show actions in Toolbar	Select this check box to have buttons that invoke XPath-related actions displayed on the Main Toolbar .
Show actions in Main Menu	When this check box is selected, XPath-related actions are available from the main menu.
Colors	In this area, configure color indication during execution of XPath expressions. Clicking on the color box will open the Select Color dialog in which you can modify the current color indication. – Highlight Color - in this area, select the color to indicate XPath matches in the editor. – Context Node Color - in this area, select the color to indicate the current context node.

Framework Integration

File | Settings | Languages and Frameworks | PHP | Framework Integration for Windows and Linux

PhpStorm | Preferences | Languages and Frameworks | PHP | Framework Integration for OS X

`Ctrl+Alt+S`

The page is available only when the PHP and the Framework MVC Structure Support plugins are enabled. The plugins are bundled with PhpStorm and activated by default. If the plugins are not activated, enable them on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

Use this page to enable viewing the structure of your framework-specific project in the terms of the Model-View-Controller pattern .

PhpStorm parses the structure of a framework-specific project tree based on the following assumptions:

1. According to the framework specification, a project should have a directory with application configuration files, a `web` directory with `js` and `css` files, and directory with a number of modules. The structure of the application `config` file and `web` directories is predefined and cannot be customized, therefore PhpStorm can parse such directories itself, you do not need to provide any additional information.
2. However inside the project, the directories can be arranged in any way and can be named without any restrictions. Moreover, each module should contain some specific folders that can be located anywhere inside the module and can have any name. Therefore, PhpStorm cannot detect the type of such folders itself.

To enable the MVC view of the project, you need to mark application roots and module roots and mark the project elements as various framework-specific types.

Item	Description
Framework name	In this drop-down list, specify the framework your project is based on. The available options are: – Symfony2 – Yii
File/Directory	In this field, select the file or folder to associate with a framework-specific type.

Element	In this field, specify the framework-specific type of element for the selected file or folder. The available options for Symfony2 are:
	– Resources
	– app for directories with application settings
	– src
	– vendor
	– web for web roots
	– Bundle root: choose this option to mark a directory as the root of a module or a Symfony bundle. Modules/bundles usually have their own types inside, so subdirectories under a bundle root can be marked as:
	  – Controller
	  – Dependency/Injection
	  – Form
	  – Tests
	
	The available options for Yii are:
	– <none>
	– Application base directory
	– Assets
	– Controllers
	– Extensions
	– Layouts
	– Models
	– Runtime directory
	– System view directory
	– Themes
	– Views
	– Module root :
	  – Components
	  – Controllers
	  – Extensions
	  – Messages
	  – Models
	  – Views

Keyboard Shortcuts and Mouse Reference

Using shortcuts is a major way to maximum efficiency and productivity of PhpStorm. This part lists keystroke combinations and their functions for the Default keymap defined in Keymap dialog box.

Warning! The key combinations documented in this part may fail to perform the function described, if you are using a customized keymap .

To get a printable copy of the default keymap, choose Help | Default Keymap Reference on the main menu.

Tip Note that in certain operating systems the key and mouse combinations may not work as described here. In this case, it's necessary to tweak the operating system's keymap. For example, if you are using Ubuntu, mind the windows manager, whose shortcuts conflict with that of PhpStorm.

In this part:

- Keyboard Shortcuts by Category
- Keyboard Shortcuts by Keystroke

Keyboard Shortcuts By Category

In this part you can find reference information about the keyboard shortcuts grouped by functional categories:

- Advanced Editing
- Basic Editing
- Code Folding
- Running and Debugging
- General
- Search
- Navigation Between Bookmarks
- Navigation Between IDE Components
- Navigation in Source Code
- Refactoring

Advanced Editing

FunctionShortcutUse this shortcut to...		
Comment with Line Comment	Ctrl+Slash	Comment/uncomment current line or selected block with line comments.
Comment with Block Comment	Ctrl+Shift+Slash	Comment/uncomment code with block comments.
Quick Documentation Look-up	Ctrl+Q	Show a pop-up window with PHPDoc or JSDoc comments for the symbol at the caret.
Show Table Data	Ctrl+Q	Show a pop-up window that displays the `create table` query for the database table at the caret and the first 10 rows of the table.
Pop-up Hector	Ctrl+Shift+Alt+H	Show the Highlighting level pop-up window to configure highlighting in the current file.
Parameter Info	Ctrl+P	Show parameters of the method call at the caret.
Context Info	Alt+Q	Show the current method or class declaration when it is not visible.
Error Description	Ctrl+F1	Show an error or warning description at the caret.
External Documentation	Shift+F1	Open browser with the documentation for the selected item.
Override Methods...	Ctrl+O	Override base class methods in the current class.
Surround with...	Ctrl+Alt+T	Surround selected code fragment with `if`, `do`, tags or other constructs.
Basic Code Completion	Ctrl+Space Alt+Slash	Code completion for any class, method or variable.
SmartType Code Completion	Ctrl+Shift+Space	Code Completion filtering the lookup list basing on expected type.
Expand Word	Alt+Slash	Goes through the names of classes, methods, keywords and variables in the current visibility scope.
Insert Live Template...	Ctrl+J	Show a pop-up list of starting with a specified prefix.
Surround with Live Template...	Ctrl+Alt+J	Surround the selection with one of the templates.
Next Template Variable	Tab	In templates: move the caret to the next template variable.
Previous Template Variable	Shift+Tab	In templates: move the caret to the previous template variable.

Basic Editing

Function	Shortcut	Use this shortcut to...
Enter	`Enter`	Depending on the context: – In a lookup list: select an item. – In the editor: enter a new line and set the caret at its beginning.
Tab	`Tab`	In the editor: – With selection: indent selected lines. – Without selection: insert a tab symbol (or corresponding number of space characters). In a lookup list: – No code after the caret: select an item. – Some code after the caret: select an item and substitute the code after the caret with it.
Delete	`Delete`	Depending on the context: – In the editor: delete selected symbol/block. – In a usage view: exclude a selected item. – In other views: remove selected items.
Backspace	`Backspace`	Delete a character to the left of the caret.
Undo	`Ctrl+Z`	Undo last operation.
Redo	`Ctrl+Shift+Z`	Redo last undone operation.
Cut	`Ctrl+X`	Cut a current line or a selected code block to the Clipboard.
Copy	`Ctrl+C`	Copy a current line or a selected code block to the Clipboard.
Paste	`Ctrl+V`	Paste from the Clipboard to the caret location.
Paste from History	`Ctrl+Shift+V`	Paste selected entry from the Clipboard to the caret location.
Up	`Up`	Move the caret one line up.
Up with Selection	`Shift+Up`	Move the caret one line up selecting the text.
Down	`Down`	Move the caret one line down.
Down with Selection	`Shift+Down`	Move the caret one line down selecting the text.
Left	`Left`	Move the caret one character to the left.
Left with Selection	`Shift+Left`	Move the caret one character to the left selecting the text.
Right	`Right`	Move the caret one character to the right.
Right with Selection	`Shift+Right`	Move the caret one character to the right selecting the text.
Go to Page Bottom	`Ctrl+Page Down`	Move the caret down to the page bottom.
Go to Page Bottom with Selection	`Ctrl+Shift+Page Down`	Move the caret down to the page bottom, selecting the text.
Go to Page Top	`Ctrl+Page Up`	Move the caret up to the page top.
Go to Page Top with Selection	`Ctrl+Shift+Page Up`	Move the caret up to the page bottom, selecting the text.
Page Down	`Page Down`	Move the caret one page down.
Page Down with Selection	`Shift+Page Down`	Move the caret one page down, selecting the text.
Page Up	`Page Up`	Move the caret one page up.
Page Up with Selection	`Shift+Page Up`	Move the caret one page up, selecting the text.
Scroll Down	`Ctrl+Down`	Scroll the text one line down.
Scroll to Center	`Ctrl+M`	Scroll a line at caret to the center of the screen.
Scroll Up	`Ctrl+Up`	Scroll the text one line up.
Move to Line End	`End`	Move the caret to the end of line.
Move to Line End with Selection	`Shift+End`	Move the caret to the end of line, selecting the text.
Move to Line Start	`Home`	Move the caret to the beginning of line.
Move to Line Start with Selection	`Shift+Home`	Move the caret to the beginning of line, selecting the text.
Move to Next Word	`Ctrl+Right`	Move the caret to the next word.
Move to Next Word with Selection	`Ctrl+Shift+Right`	Move the caret to the next word, selecting it.
Move to Previous Word	`Ctrl+Left`	Move the caret to the previous word.
Move to Previous Word with Selection	`Ctrl+Shift+Left`	Move the caret to the previous word, selecting it.
Move to Text End	`Ctrl+End`	Move the caret to the end of text.
Move to Text End with Selection	`Ctrl+Shift+End`	Move the caret to the end of text, selecting it.
Move to Text Start	`Ctrl+Home`	Move the caret to the beginning of text.
Move to Text Start with Selection.	`Ctrl+Shift+Home`	Move the caret to the beginning of text, selecting it.
Select All	`Ctrl+A`	Select the entire text opened in the editor.

Delete Line at Caret	`Ctrl+Y`	Delete the line where the caret is currently located.
Delete to Word End	`Ctrl+Delete`	Delete the word starting from the current caret location up to the word end.
Delete to Word Start	`Ctrl+Backspace`	Delete the word starting from the current caret location up to the word start.
Toggle Insert/Overwrite	`Insert`	Toggle insert/overwrite modes.
Duplicate Line or Block	`Ctrl+D`	Duplicate selected block or the line at the caret.
Toggle Case	`Ctrl+Shift+U`	Toggle case of the selected text block.
Move to Code Block End	`Ctrl+Close Bracket`	Move the caret to the current code block end, highlighting the block limits.
Move to Code Block End with Selection	`Ctrl+Shift+Close Bracket`	Move the caret to the current code block end, selecting the code beginning from the initial caret location.
Move to Code Block Start	`Ctrl+Open Bracket`	Move the caret to the current code block start, highlighting the block limits.
Move to Code Block Start with Selection	`Ctrl+Shift+Open Bracket`	Move the caret to the current code block start, selecting the code beginning from the initial caret location.
Start New Line	`Shift+Enter`	Start a new line after the current one positioning the caret in accordance with the current indentation level.
Start New Line Before Current One	`Ctrl+Alt+Enter`	Start a new line before the current one.
Join Lines	`Ctrl+Shift+J`	Concatenate the selected lines into one or concatenate the line where the caret is currently located with the next line.
Split Line	`Ctrl+Enter`	Split the selected line at the point where the caret is located, leaving the caret at the end of the first line.
Select Word at Caret	`Ctrl+W`	Select successively increasing code blocks starting from the current caret location.
Unselect Word at Caret	`Ctrl+Shift+W`	Remove sequentially the selection made by the action.
Indent Selection	`Tab`	Move the selected block to the next indentation level.
Unindent Selection	`Shift+Tab`	Move the selected block to the previous indentation level.
Auto-Indent Lines	`Ctrl+Alt+I`	Indent the current line or selected block according to the Code Style settings.

Code Folding

Command	Shortcut	Description
Expand	`Ctrl+NumPad Plus`	Expand the current collapsed fragment
Collapse	`Ctrl+NumPad -`	Collapse the current folding region
Expand Recursively	`Ctrl+Alt+NumPad Plus`	Expand the current folded fragment and all the subordinate collapsed folding regions within that fragment
Collapse Recursively	`Ctrl+Alt+NumPad -`	Collapse the current folding region and all the subordinate folding regions within it
Expand All	`Ctrl+Shift+NumPad Plus`	Expand all collapsed fragments within the selection, or, if nothing is selected, expand all the collapsed fragments in the current file
Collapse All	`Ctrl+Shift+NumPad -`	Collapse all folding regions within the selection, or, if nothing is selected, collapse all the folding regions in the current file

Expand to level | 1, 2, 3, 4 or 5	`Ctrl+NumPad *, 1`
	`Ctrl+NumPad *, 2`
	`Ctrl+NumPad *, 3`
	`Ctrl+NumPad *, 4`
	`Ctrl+NumPad *, 5`
		Expand the current fragment and all the nested fragments up to the specified level

Expand all to level | 1, 2, 3, 4 or 5	`Ctrl+Shift+NumPad *, 1`
	`Ctrl+Shift+NumPad *, 2`
	`Ctrl+Shift+NumPad *, 3`
	`Ctrl+Shift+NumPad *, 4`
	`Ctrl+Shift+NumPad *, 5`
		Expand all the collapsed fragments in the file up to the specified nesting level

Fold Selection / Remove region	`Ctrl+Period`	Collapse the selected fragment and create a custom folding region for it to make it "foldable" / Expand the current fragment and remove the corresponding custom folding region to make the fragment "unfoldable"

Running and Debugging

Function	Shortcut	Use this shortcut to...
Run	Shift+F10	Run a program.
Choose configuration and run	Shift+Alt+F10	Quickly select run/debug configuration and run or edit it.
Rerun	Ctrl+F5	Repeat execution with the same settings, with the same tab of the Run tool window having the focus.
Rerun without loosing the focus in the editor	Shift+F10	Repeat execution with the same settings, with the same tab of the editor having the focus.
Debug	Shift+F9	Debug a program.
Choose configuration and debug	Shift+Alt+F9	Quickly select run/debug configuration and debug or edit it.
Update running application	N/A	Reload updated resources and recompiled classes of the currently running application.
Step Over	F8	Step to the next line in the current file. See Stepping Through the Program .
Step Into	F7	Step to the next executed line. See Stepping Through the Program .
Smart Step Into	Shift+F7	Select the method to step in, if the current line contains multiple method call expressions. See Choosing a Method to Step Into .
Step Out	Shift+F8	Step to a first executed line after returning from the current method. See Stepping Through the Program .
Force Step Over	Shift+Alt+F8	Run until the next line in this method or file, skipping the methods referenced at the current execution point and ignoring breakpoints. See Stepping Through the Program .
Force Step Into	Shift+Alt+F7	Steps into the method called in the current execution point even if this method is to be skipped. See Stepping Through the Program .
Run to Cursor	Alt+F9	Run to the line where the caret is located. See Stepping Through the Program .
Force Run To Cursor	Ctrl+Alt+F9	Run to the line where the caret is located, ignoring breakpoints. See Stepping Through the Program .
Resume Program	F9	Resume program execution.
Stop Program	Shift+F2	Terminate a debugging session.
Evaluate Expression	Alt+F8	Evaluate an arbitrary expression.
Quick Evaluate Expression	Ctrl+Alt+F8	Evaluate an arbitrary expression without calling Evaluate Expression dialog.
Toggle Breakpoint	Ctrl+F8	Toggle breakpoint at the current line.
View Breakpoints	Ctrl+Shift+F8	View/manage all breakpoints.

General

Function	Shortcut	Use this shortcut to...
Close Active Tab	Ctrl+Shift+F4	Close an active tab in a tool window (for example, Find tool window).
Close Editor	Ctrl+F4	Close an active editor.
Edit Source	F4	Open an editor for the selected item or items and give focus to the last opened file.
Escape	Escape	Depending on the context: – In the editor: close pop-up windows, terminate search, or remove highlighting. – In a tool window: return focus to the editor.
Export to Text File	Alt+O	Export contents of a tool window to a text file. This feature applies to the Version Control Tool Window , Messages Tool Window , and other tool windows that provide the export button on the window toolbar.
New...	Alt+Insert	Create a new class, file or directory. See Populating Modules .
Save All	Ctrl+S	Save all files and settings.
Select Next Tab	Alt+Right	When several tabs are open in the editor or a view, open the next tab to the right (or first tab if the current one is the last).
Select Previous Tab	Alt+Left	When several tabs are open in the editor or a view, open the next tab to the left (or last tab if the current one is the first).
Show Intention Action	Alt+Enter	Display intention actions (if any) for the code where the caret is currently located, or the selected GUI component in a form.
Synchronize	Ctrl+Alt+Y	Detect all externally changed files and reload them from disk.

Search

Function	Shortcut	Use this shortcut to...
Find	Ctrl+F	Initiate text search .
Replace	Ctrl+R	Initiate text search and replace .
Search for next/ previous occurrence	F3 / Shift+F3	Navigate to the next/previous occurrence of a selected word in the editor.
Find Word at Caret	Ctrl+F3	Search in the editor for the word where the caret is currently located.
Incremental Search	Ctrl+F	Initiate text search .
Find in Path	Ctrl+Shift+F	Initiate search for a text string in the specified scope .
Replace in Path	Ctrl+Shift+R	Initiate search and replace in the specified scope .
Find Usages	Alt+F7	Initiate search for usages of the selected symbol in the specified scope .
Find Usages in File	Ctrl+F7	Initiate search for usages of the selected symbol in the current file .
Highlight Usages in File	Ctrl+Shift+F7	Highlight usages of a symbol at caret.
Show Usages	Ctrl+Alt+F7	Show usages of a symbol at caret in a pop-up window. Use list of found usages to jump to the desired location.
Find Action	Ctrl+Shift+A	Find an action, bypassing menus. See Finding Actions .

Navigation Between Bookmarks

Function	Shortcut	Use this shortcut to...
Go to Bookmark <number>	`Ctrl+Number`	Navigate to a numbered bookmark with the corresponding number.
Toggle Bookmark	`F11`	Turn anonymous bookmark on or off.
Toggle Bookmark with Mnemonic	`Ctrl+F11`	Turn bookmark with mnemonic on or off.
Show Bookmarks	`Shift+F11`	Open Bookmarks dialog to manage existing bookmarks and navigate between them.

Navigation Between IDE Components

In this section you can find keyboard shortcuts for navigation between:

- Views and Windows .
- Differences .

Views and Windows

Function	Shortcut	Use this shortcut to...
Select Target	Alt+F1	Move focus from the current file, class, method or reference to a data source, to a view suggested in the Select Target pop-up menu. Refer to Navigating Between IDE Components .
Collapse all	Ctrl+NumPad -	Collapse all nodes in a tree view.
Expand all	Ctrl+NumPad Plus	Expand all nodes in a tree view.
Switcher	Ctrl+Tab	Navigate between files opened in the editor, and tool windows.
Open tool window	Alt+Number	Open a tool window with the specified number.
Hide Active Window	Shift+Escape	Hide the currently active tool window.
Jump to Last Window	F12	Activate the last focused tool window.

Differences

Function	Shortcut	Use this shortcut to...
Move to Next Difference	F7	Navigate to the next difference in view .
Move to Previous Difference	Shift+F7	Navigate to the previous difference in view .

Navigation In Source Code

Function	Shortcut	Use this shortcut to...
File Structure Pop-up	`Ctrl+F12`	Display the Structure pop-up window for quick navigation through the current file.
Select target	`Alt+F1`	Move focus from the current file, class, method or reference to a data source table to a view suggested in the Select Target pop-up menu. See Navigating Between IDE Components .
Recent Files	`Ctrl+E`	Show the list of recently opened files .
Recently Changed Files	`Ctrl+Shift+E`	Show the list of recently updated files .
Type Hierarchy	`Ctrl+H`	Browse hierarchy for the selected class .
Navigate to Class	`Ctrl+N`	Navigate directly to a class in project by specifying its name in a pop-up dialog box.
Navigate to File	`Ctrl+Shift+N`	Navigate directly to a file in project by specifying its name in a pop-up dialog box.
Navigate to Recently Opened File	`Ctrl+E`	Show the list of recently opened files .
Navigate to Recently Changed File	`Ctrl+Shift+E`	Show the list of recently updated files .
Navigate to Line	`Ctrl+G`	Navigate to any line in the current file by specifying its number.
Navigate to Declaration	`Ctrl+B`	Navigate to declaration of a symbol at caret.
Navigate to Implementation	`Ctrl+Alt+B`	Navigate to implementation of the item at caret. This type of navigation lets you jump from a base type or member to any of its end implementations skipping the intermediate steps in the inheritance chain. For example, suppose that class `MyClass` inherits abstract class `MyAbstractClass` , which, in its turn, implements interface `MyOriginalInterface` . Choosing Go to Implementation when the caret is positioned at `MyOriginalInterface` takes you directly to the declaration of `MyClass` .
Navigate to Type Declaration	`Ctrl+Shift+B`	Navigate to a type declaration of a symbol at caret, the symbol being a variable or a method call.
Navigate to Super Method	`Ctrl+U`	Navigate to a super method declaration of a method under the caret.
Navigate to Test/Test Subject	`Ctrl+Shift+T`	Navigate to a test for the class at caret, if any, or navigate from a test to a test subject.
Navigate to Related Symbol	`Ctrl+Alt+Home`	Navigate between files with complicated relationships between them.
Navigate to Next Method	`Alt+Down`	Navigate to the next method declaration in the active editor tab.
Navigate to Previous Method	`Alt+Up`	Navigate to the previous method declaration in the active editor tab.
Navigate to Opening Brace	`Ctrl+Open Bracket`	Navigate to the start of the current code block.
Navigate to Closing Brace	`Ctrl+Close Bracket`	Navigate to the end of the current code block.
Back	`Ctrl+Alt+Left`	Undo last navigation operation. Note On an OS X computer, you can also use the three-finger right-to-left swipe gesture.
Forward	`Ctrl+Alt+Right`	Redo last undone navigation operation. Note On an OS X computer, you can also use the three-finger left-to-right swipe gesture.
Navigate to Previous Occurrence	`Ctrl+Alt+Up`	Navigate to a previous found item.
Navigate to Next Occurrence	`Ctrl+Alt+Down`	Navigate to a next found item.
Last Edit Location	`Ctrl+Shift+Backspace`	Move through the most recent change points.
Navigate to Next Highlighted Error	`F2`	Navigate to the next found error/warning.
Navigate to Previous Highlighted Error	`Shift+F2`	Navigate to the previous found error/warning.

Refactoring

FunctionShortcut	Use this shortcut to...	
Rename	`Shift+F6`	Rename the selected file, class, field, method, etc. and change all references to it accordingly.
Change Method Signature	`Ctrl+F6`	Change the signature of the selected method and update all the corresponding method calls.
Move	`F6`	Move the selected class, package or static member to another package or class and update all the corresponding references.
Copy	`F5`	Create a copy of the selected class, file or directory in the same or different directory or package.
Clone	Create a copy of the selected class in the same package.	
Safe Delete	`Alt+Delete`	Delete the selected class, method or field checking its usages.
Extract Method	`Ctrl+Alt+M`	Turn the selected code fragment into a method.
Extract Variable	`Ctrl+Alt+V`	Create a new variable and use the selected expression as its value.
Extract Field	`Ctrl+Alt+F`	Create a new field and use the selected expression as its value.
Extract Constant	`Ctrl+Alt+C`	Create a new constant (static final field) and use the selected expression as its value.
Extract Parameter	`Ctrl+Alt+P`	Turn the selected expression into a new method parameter.
Inline	`Ctrl+Alt+N`	Inline the selected method or variable.

Keyboard Shortcuts By Keystroke

In this part you can find reference information on keyboard shortcuts grouped by keystroke:

- Alt
- Alt+Shift
- Ctrl
- Ctrl+Alt
- Ctrl+Shift
- Function Keys
- Insert, Delete and Navigation Keys
- Shift
- Ctrl+Alt+Shift

To view a full list of available shortcuts, navigate to File | Settings and click Keymap under IDE Settings.

Alt

This section lists and describes the keyboard shortcuts that include the `Alt` key:

- Alt+Alphanumeric keys
- Alt+Navigation keys
- Alt+Function (F) keys

Alt+Alphanumeric keys

Shortcut	Function	Use this shortcut to...
`Alt+O`	Export to Text File	Export a tool window's content to a text file.
`Alt+Q`	Context Info	Show the current method or class declaration when it is not visible.
`Alt+Number`	Open tool window	Open a tool window with the corresponding number.
`Alt+Slash`	Code completion / Expand word	Expand string at caret to any word in the visible scope that starts with the same characters.
`Alt+Back Quote`	VCS operations	Show quick list with the most required version control commands.

Alt+Navigation keys

Shortcut	Function	Use this shortcut to...
`Alt+Delete`	Safe Delete	Delete selected class/method/field checking its usages.
`Alt+Enter`	Show Intention Action	Display intention actions (if any) for a code where the caret is currently located.
`Alt+Home`	Activate Navigation Bar	Bring focus to the Navigation bar .
`Alt+Insert`	Create new entity	Depending on the context: – In the navigation views: create a new class, file or directory , using the New pop-up menu.
`Alt+Down`	Navigate to Next Method	Navigate to the next method declaration in the active editor tab.
`Alt+Left`	Select Previous Tab	Depending on the context: – When several tabs are opened in the editor or a view, open the next tab to the left (or the last tab if the current one is the first). – In the Differences Viewer for Files invoked from the Update Project Info tab of the Version Control tool window, compare the local copy of the previous file with its update from the server.
`Alt+Right`	Select Next Tab	Depending on the context: – When several tabs are opened in the editor or a view, open the next tab to the right (or the first tab if the current one is the last). – In the Differences Viewer for Files invoked from the Update Project Info tab of the Version Control tool window, compare the local copy of the next file with its update from the server.
`Alt+Up`	Navigate to Previous Method	Navigate to the previous method declaration in the active editor tab.

Alt+Function (F) keys

Shortcut	Function	Use this shortcut to...
`Alt+F1`	Select Target	Move focus from the current file, class, method or reference to a data source table to a view suggested in the Select Target pop-up menu. See Navigating Between IDE Components .
`Alt+F7`	Find Usages	Initiate search for usages .
`Alt+F8`	Evaluate Expression	Debugger: Evaluate an arbitrary expression .
`Alt+F9`	Run to Cursor	Debugger: Run to the line where the caret is located.

Alt+Shift

This section lists and describes the keyboard shortcuts that include the `Shift+Alt` keys:

Shortcut	Function	Use this shortcut to...
`Shift+Alt+F7`	Force Step Into	Step into the method called in the current execution point, even if this method is to be skipped.
`Shift+Alt+F8`	Force Step Over	Run until the next line in this method or file, skipping the methods referenced at the current execution point and ignoring breakpoints.
`Shift+Alt+F9`	Debug	Quickly select run/debug configuration and debug/edit it.
`Shift+Alt+F10`	Run	Quickly select run/debug configuration and run/edit it.

Ctrl

This section lists and describes the keyboard shortcuts that include the `Ctrl` key:

- Ctrl+Alphanumeric keys
- Ctrl+Navigation keys
- Ctrl+Symbol keys
- Ctrl+Numpad keys
- Ctrl+Function (F) keys

Ctrl+Alphanumeric keys

Shortcut	Function	Use this shortcut to...
`Ctrl+A`	Select All	Select the entire text in the active editor.
`Ctrl+B`	Navigate to Declaration	Navigate directly to an element's declaration from any usage.
`Ctrl+C`	Copy	Copy selected text to the Clipboard.
`Ctrl+D`	Duplicate Line or Block	Duplicate selected block or line at caret.
`Ctrl+E`	Recent Files Recent find usages	Show the list of recently opened files . When the Find tool window has the focus, use this shortcut to show the list of recent find usages results .
`Ctrl+F`	Find	Initiate text search in the editor.
`Ctrl+G`	Navigate to Line	Navigate the to a line with the specified number in the current file.
`Ctrl+H`	Type Hierarchy	Browse hierarchy for the selected class.
`Ctrl+J`	Insert Live Template	Show a pop-up list of Live Templates starting with a specified prefix.
`Ctrl+M`	Scroll to Center	Scroll a line at caret to the center of the screen.
`Ctrl+N`	Navigate to Class	Jump to a class in the project with the specified name.
`Ctrl+O`	Override Methods	Override base class methods in the current class.
`Ctrl+P`	Parameter Info	Show parameters of the method call at the caret.

`Ctrl+Q`
 	
`Alt+Mouse Button2`
	Quick documentation	Show a pop-up window with documentation for the symbol at caret.
		In the Database tool window : show a pop-up window that displays the `create table` query for the database table at the caret and the first 10 rows of the table.

`Ctrl+R`	Replace	Call the Replace Text dialog box.
`Ctrl+S`	Save All	Save all files and settings.
`Ctrl+U`	Navigate to Super Method	Navigate to a super method declaration of a method at caret
`Ctrl+W`	Select Word at Caret	Successively select expanding blocks of text, starting from the word at caret. (Use this shortcut repeatedly to select expressions.)
`Ctrl+Y`	Delete Line at Caret	Delete a word starting from the current caret location up to the end of word.
`Ctrl+Z`	Undo	Undo last operation.
`Ctrl+Shift+Z`	Redo	Redo last undone operation.
`Ctrl+Number`	Navigate to bookmark	Navigate to a numbered bookmark with corresponding number.

Ctrl+Navigation keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Tab`	Switcher	Navigate between the files opened in the editor, and tool windows.
`Ctrl+Backspace`	Delete to Word Start	Delete a word starting from the current caret location up to the word start.
`Ctrl+Delete`	Delete to Word End	Delete a word starting from the current caret location up to the word end.
`Ctrl+End`	Move to Text End	Move the caret to the end of text.
`Ctrl+Enter`	Split Line or Open Item	Depending on the context: – In the editor: Intelligently split the current line into 2 lines, shifting quotes, etc. as necessary. – In the Tool Windows: Open an Editor tab or tabs for the selected item or items, respectively.
`Ctrl+Home`	Move to Text Start	Jump to the beginning of the text.
`Ctrl+C`	Copy	Copy a current line or a selected code block to the Clipboard.
`Ctrl+Space`	Basic Code Completion	Complete code for any class, method or variable.
`Ctrl+Page Down`	Navigate to Page Bottom	Move the caret down to the page bottom.

Ctrl+Page Up	Navigate to Page Top	Move the caret up to the page top.
Ctrl+Down	Scroll Down	Move line at caret one down, preserving syntactical correctness.
Ctrl+Left	Move to Previous Word	Move the caret to the previous word.
Ctrl+Right	Move to Next Word	Move the caret to the next word.
Ctrl+Up Ctrl+Shift+Up	Scroll Up	Move line at caret one up, preserving syntactical correctness.
Ctrl + Ctrl + Home / End		Select text from the caret position to the beginning/end of the current line.

Ctrl+Symbol keys

Shortcut	Function	Use this shortcut to...
Ctrl+Open Bracket	Move to Code Block Start	Move the caret to the beginning of the current code block, highlighting its limits.
Ctrl+Close Bracket	Move to Code Block End	Move the caret to the end of the current code block, highlighting its limits.
Ctrl+Slash Ctrl+Numpad/	Comment with Line Comment	Comment/uncomment current line or selected block with line comments.
Ctrl+= Ctrl+NumPad Plus	Expand All	Expand all folding blocks.
Ctrl+NumPad -	Collapse All	Collapse all folding blocks.

Ctrl+Numpad keys

Shortcut	Function	Use this shortcut to...
Ctrl+Numpad/ Ctrl+Slash	Comment with Line Comment	Comment/uncomment current line or selected block with line comments.
Ctrl+NumPad Plus Ctrl+=	Expand All	Expand all folding blocks.
Ctrl+NumPad -	Collapse All	Collapse all folding blocks.

Ctrl+Function (F) keys

Shortcut	Function	Use this shortcut to...
Ctrl+F1	Error Description	Show an error or warning description at the caret.
Ctrl+F3	Find Word at Caret	Search in the editor for the word where the caret is currently located.
Ctrl+F6	Change Method Signature	Refactor a selected method signature and update all references.
Ctrl+F7	Find Usages in File	Initiate search for usages .
Ctrl+F8	Toggle Breakpoint	Toggle breakpoint at caret.
Ctrl+F9	Make Project	Compile all modified and dependent files in a project.
Ctrl+F11	Toggle Bookmark with mnemonic.	Turn bookmark with mnemonic on or off.
Ctrl+F12	File Structure Pop-up	Show the current file structure in the File Structure pop-up window for quick navigation.

Ctrl+Alt

This section lists and describes the keyboard shortcuts that include the `Ctrl+Alt` keys:

- Ctrl+Alt+Alphanumeric keys
- Ctrl+Alt+Navigation keys
- Ctrl+Alt+Function (F) keys

Ctrl+Alt+Alphanumeric keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Alt+B`	Navigate to Implementation	Navigate to implementation of an item at the caret. This type of navigation lets you jump from a base type or member to any of its end implementations skipping the intermediate steps in the inheritance chain. For example, suppose that class `MyClass` inherits abstract class `MyAbstractClass`, which, in its turn, implements interface `MyOriginalInterface`. Choosing Go to Implementation when the caret is positioned at `MyOriginalInterface` takes you directly to the declaration of `MyClass`.
`Ctrl+Alt+C`	Extract Constant	Replace selected expression with a constant (static final field) (Refactoring).
`Ctrl+Alt+F`	Extract Field	Put a selected expression result into a field (Refactoring).
`Ctrl+Alt+H`	Call Hierarchy	Browse call hierarchy for the selected method.
`Ctrl+Alt+I`	Auto-indent Lines	Indent current line or selected block according to the Code Style Settings.
`Ctrl+Alt+J`	Surround with Live Template	Surround the selection with one of the Live Templates.
`Ctrl+Alt+M`	Extract Method	Create a method from the selected code (Refactoring).
`Ctrl+Alt+N`	Inline	Inline a selected method/variable (Refactoring).
`Ctrl+Alt+P`	Extract Parameter	Turn a selected expression into a method parameter (Refactoring).
`Ctrl+Alt+T`	Surround with	Surround selected code fragment with if, while, try/catch, or another construct.
`Ctrl+Alt+V`	Extract Variable	Put selected expression result into a variable (Refactoring).
`Ctrl+Alt+Y`	Synchronize	Detect all externally changed files and reload them from disk.

Ctrl+Alt+Navigation keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Alt+Enter`	Start new line before current one	Start a new line before the current one.
`Ctrl+Alt+Down` `Ctrl+Alt+Up`	Navigate to Next/Previous Occurrence	Navigate to a next found item .
`Ctrl+Alt+Left`	Back	Undo last navigation operation. Note On an OS X computer, you can also use the three-finger right-to-left swipe gesture.
`Ctrl+Alt+Right`	Forward	Redo last undone navigation operation. Note On an OS X computer, you can also use the three-finger left-to-right swipe gesture.
`Ctrl+Alt+Home`	Navigate to Related Symbol	Navigates between files with the various relationships. See Navigation in Source Code .

Ctrl+Alt+Function (F) keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Alt+F6`	Switch to another coverage suite.	Open the Coverage Suites popu-up window and select the desired suite to run.
`Ctrl+Alt+F7`	Show usages	Show usages of a symbol at the caret.
`Ctrl+Alt+F8`	Quick Evaluate Expression	Evaluate an arbitrary expression without calling Evaluate Expression dialog box.
`Ctrl+Alt+F9`	Force Run To Cursor	Run to the line where the caret is located, ignoring breakpoints.

Ctrl+Shift

This section lists and describes the keyboard shortcuts that include the `Ctrl+Shift` keys:

- Ctrl+Shift+Alphanumeric keys
- Ctrl+Shift+Navigation keys
- Ctrl+Shift+Symbol keys
- Ctrl+Shift+Numpad keys
- Ctrl+Shift+Function (F) keys

Ctrl+Shift+Alphanumeric keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+A`	Find Action	Find an action, bypassing menus. See Finding Actions .
`Ctrl+Shift+B`	Navigate to Type Declaration	Navigate to type declaration of a variable or a method call at caret.
`Ctrl+Shift+E`	Navigate to Recently Changed File	Show the list of recently updated files .
`Ctrl+Shift+F`	Find in Path	Initiate text search in the specified path .
`Ctrl+Shift+H`	Method Hierarchy	Browse hierarchy for the selected class.
`Ctrl+Shift+N`	Navigate to File	Jump to the specified file in project.
`Ctrl+Shift+R`	Replace in Path	Initiate text replacement in the specified path .
`Ctrl+Shift+T`	Navigate to Test/Test Subject	Navigate to a test for the class at caret, if any, or create a new test class. For a test class, navigate to its test subject.
`Ctrl+Shift+U`	Toggle Case	Toggle case of the selected text fragment.
`Ctrl+Shift+V`	Paste from History	Paste from recent Clipboards.
`Ctrl+Shift+W`	Deselect Word at Caret	Remove sequential selection made by the Select Word at Caret action .
`Ctrl+Shift+Z`	Redo	Redo the last Undo operation .

Ctrl+Shift+Navigation keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+End`	Move to Text End with Selection	Select text from the current caret position to the end of text, and move caret to the end of text.
`Ctrl+Shift+Home`	Move to Text Start with Selection	Select text from the current caret position to the start of text, and move caret to the start of text.
`Ctrl+Shift+Right`	Move to Word End with Selection	Select text from the current caret position to the end of word, and move caret to the end of word.
`Ctrl+Shift+Left`	Move to Word Start with Selection	Select text from the current caret position to the beginning of the current word, and move caret to to the beginning of this word.
`Ctrl+Shift+V`	Paste from History	Paste from recent Clipboards.
`Ctrl+Shift+Space`	SmartType Code Completion	Complete code, filtering the lookup list based on an expected type.
`Ctrl+Shift+Page Down`	Navigate to Page Bottom with Selection	Move the caret down to the page bottom selecting the text.
`Ctrl+Shift+Page Up`	Navigate to Page Top with Selection	Move the caret up to the page bottom selecting the text
`Ctrl+Shift+Down`	Move Line Down	Move line at caret one down, preserving syntactical correctness.
`Ctrl+Shift+Up`	Move Line Up	Move line at caret up, preserving syntactical correctness.
`Ctrl+Shift+Backspace`	Last Edit Location	Jump to the place of the last editing.

Ctrl+Shift+Symbol keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+Open Bracket`	Move to Code Block Start with Selection	Move the caret to the beginning of the current code block, selecting the code from the initial caret location.
`Ctrl+Shift+Close Bracket`	Move to Code Block End with Selection	Move the caret to the end of the current code block, selecting the code from the initial caret location.
`Ctrl+Shift+Slash`	Comment with Block Comment	Comment/uncomment code with block comments.
`Ctrl+NumPad Plus`	Expand All	Expand all folding blocks.
`Ctrl+Shift+NumPad -`	Collapse All	Collapse all folding blocks.

Ctrl+Shift+Numpad keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+Numpad/`	Comment with Block Comment	Comment/uncomment code with block comments.
`Ctrl+Shift+Numpad+`	Expand All	Expand all folding blocks.
`Ctrl+NumPad Plus`		

`Ctrl+Shift+Numpad-`	Collapse All	Collapse all folding blocks.

Ctrl+Shift+Function (F) keys

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+F4`	Close Active Tab	Close an active tab in a tool window
`Ctrl+Shift+F7`	Highlight Usages in File / Highlight Method Exit Points	Highlight usages of a symbol where the caret is currently located. If the caret is placed on one of the method's exit points, like `return`, all method exit points are highlighted.
`Ctrl+Shift+F8`	View Breakpoints	View/manage all breakpoints/watchpoints

Function Keys

This section describes default mappings for the function (F) keys.

Shortcut	Function	Use this shortcut to...
F1	Help	Invoke reference page.
F2	Activate in-place editing	In a GUI Designer form, enable in-place editing of the name of a selected UI component.
F3 Shift+F3	Search for next/previous occurrence	Navigate to the next/previous occurrence of a selected word in the editor.
F4	Edit Source	Depending on the context: – In Tool Windows: Open an Editor tab or tabs for the selected item or items (including GUI forms), and give focus to the last opened file.
F5	Copy	Create a copy of a selected class/file/directory in the same or a different package.
F6	Move	Move a selected class/package/static member to another package/class and correct all references.
F7	Step Into	Step to the next executed line (during debugging).
F8	Step Over	Step to the next line in the current file (during debugging).
F9	Resume Program	Resume program execution (during debugging).
F11	Toggle Bookmark	Turn anonymous bookmark on or off.
F12	Jump to Last Window	Activate a last focused tool window.

Insert, Delete and Navigation Keys

This section describes default mappings for the `Insert`, `Delete` and the navigation keys.

Shortcut	Function	Use this shortcut to...
`Delete`	Delete	Depending on the context: – In the editor: delete selected symbol/block. – In the Find tool window : exclude items from the search results. – In the Version Control tool window : delete an item from a changelist. – In other views: remove the selected item or items.
`Down`	Move down	Move the caret one line down.
`End`	Move to Line End	Move the caret to the end of line.
`Home`	Move to Line Start	Move the caret to the beginning of line.
`Insert`	Toggle Insert/Overwrite	Toggle Insert/Overwrite modes in the editor. The shape of the cursor changes according to the current mode.
`Left`	Move left	Move the caret one character to the left.
`Page Down`	Page down	Move the caret one page up.
`Page Up`	Page up	Move the caret one page up.
`Right`	Move right	Move the caret one character to the right.
`Tab`		In the editor: – With any selection: indent selected line(s). – Without any selection: insert a tab symbol (or corresponding number of space.characters). In a lookup list: – No code after the caret in the editor: select an item (like `Enter`) – Some code after the caret in the editor: select an item and replace the code after the caret with it.
`Up`	Move up	Move the caret one line up.

Shift

This section lists and describes the keyboard shortcuts that include the `Shift` key:

- Shift+Navigation keys
- Shift+Function (F) keys

Shift+Navigation keys

Shortcut	Function	Use this shortcut to...
`Shift+Down`	Down with Selection	Move the caret one line down selecting the text.
`Shift+End`	Move to Line End with Selection	Move the caret to the end of line, selecting text.
`Shift+Enter`	Start New Line	Start a new line after the current one, positioning the caret in accordance with the current indentation level (equal to sequential pressing End, Enter).
`Shift+Escape`	Hide Active Window	Hide the currently active tool window.
`Shift+Home`	Move to Line Start with Selection	Move the caret to the beginning of line, selecting the text.
`Shift+Left`	Left with Selection	Move the caret one character to the left selecting the text.
`Shift+Page Down`	Page Down with Selection	Move the caret one page down selecting the text.
`Shift+Page Up`	Page Up with Selection	Move the caret one page up selecting the text.
`Shift+Right`	Right with Selection	Move the caret one character to the right selecting the text.
`Shift+Tab`	Unindent Selection	Move selected block to the previous indent level.
`Shift+Up`	Up with Selection	Move the caret one line up selecting the text.

Shift+Function (F) keys

Shortcut	Function	Use this shortcut to...
`Shift+F1`	External Documentation	Open browser with the documentation for the selected item. Refer to Viewing Inline Documentation for details.
`Shift+F2`	One of the following: – Navigate to Previous Highlighted Error. – Stop Program.	Depending on whether you are editing or debugging: – When editing: Navigate to the previous found error/warning. – When debugging: Terminate the debugging session.
`F3` / `Shift+F3`	Search for next/previous occurrence	Jump to the next/previous occurrence of the selected word in the editor.
`Shift+F6`	Rename	Rename a statement and correct all references. (Refactoring).
`Shift+F7`	Move to Previous Difference/Smart Step Into	Move to a previous difference in a view./ Select the method to step in, if the current line contains multiple method call expressions. (Debugger).
`Shift+F8`	Step Out	Step to the first executed line after returning from a current method.
`Shift+F9`	Debug	Debug application.
`Shift+F10`	Run	Run application.
`Shift+F11`	Show Bookmarks	Open Bookmarks dialog to manage existing bookmarks and navigate between them.
`Shift+F12`	Restore Default layout	Restore the default PhpStorm layout (tool windows positions, buttons location and order). To restore the default layout, check the option Store Current Layout as Default in the Window menu.

Ctrl+Alt+Shift

This section lists and describes the keyboard shortcuts that include the `Ctrl+Shift+Alt` keys.

Shortcut	Function	Use this shortcut to...
`Ctrl+Shift+Alt+C`	Copy Reference	Copy a reference of a symbol to the Clipboard.
`Ctrl+Shift+Alt+N`	Go to Symbol	Navigate to a symbol with the specified name.
`Ctrl+Shift+Alt+H`	Pop up Hector	Open the Highlighting level pop-up window.
`Ctrl+Shift+Alt+U`	Show Uml Diagram	Open UML Class diagram for a class or package.
`Ctrl+Shift+Alt+V`	Paste Simple	Paste the last entry from the Clipboard as plain text.
`Ctrl+Shift+Alt+L`	Show Reformat File Dialog	Show reformatting dialog .
`Ctrl+Shift+Alt+I`	Run Inspection by Name	Execute an inspection by its name .
`Ctrl+Shift+Alt+Insert`	New Scratch File	Create a new scratch file with the selected language.

Tool Windows Reference

View | Tool Windows

The Tool Windows Reference contains detailed information about the functionality, controls and menus of the PhpStorm tool windows .

In this section:

- Project (Alt+1)
- Favorites (Alt+2)
- Find (Alt+3)
- Run (Alt+4)
- Debug (Alt+5)
- TODO (Alt+6)
- Structure (Alt+7)
- Hierarchy (Alt+8)
- Changes (Alt+9)
- Inspection
- Version Control
- Database
- Phing Build

Application Servers tool window

View | Tool Windows | Application Servers

For this tool window to be available, there must be a server run/debug configuration in your project , or a cloud user account must be registered in PhpStorm.

The Application Servers tool window lets you manage your applications on application servers and cloud platforms. You can start and stop server run/debug configurations and connect to cloud platforms, deploy and undeploy your application artifacts as well as perform other, associated tasks.

All the available functions are accessed by means of the toolbar icons and context menu commands.

- Icons and commands for server run configurations
- Icons and commands for server artifacts
- Icons and commands for cloud user accounts
- Icons and commands for cloud apps
- Deployment status icons

See also, Creating Run/Debug Configuration for Application Server and Working with Cloud Platforms .

Icons and commands for server run configurations

Icon	Command	Description
	Run/Connect	Start the selected run/debug configuration in the run mode. For a local configuration, normally, the corresponding server will be started. For a remote configuration, PhpStorm will connect to the server.
	Debug	Start the selected run/debug configuration in the debug mode.
	Stop/Disconnect	Stop the selected run/debug configuration. For a local configuration, normally, the corresponding server will be stopped. For a remote configuration, PhpStorm will disconnect from the server.
	Deploy All	Deploy all the artifacts associated with the selected run/debug configuration.
	Edit Configuration	Edit the settings for the selected run/debug configuration.
	Artifacts	Edit the deployment list for the selected run/debug configuration. (The Artifacts to Deploy Dialog will open.)

Icons and commands for server artifacts

Icon	Command	Description
	(Re)deploy	Deploy or redeploy the selected artifact.
	Undeploy	Undeploy the selected artifact.
	Remove	Remove the selected artifact from the corresponding deployment list and undeploy the artifact from the server.

Icons and commands for cloud user accounts

Icon	Command	Description
	Connect	Connect (log on) to the corresponding cloud platform.
	Disconnect	Disconnect (log off) from the corresponding cloud platform.
	Edit Configuration	Edit your cloud user account settings.
	Deploy	Deploy your app by means of a cloud deployment run/debug configuration.
	Debug	Deploy your app and start debugging it by means of a cloud deployment run/debug configuration.

Icons and commands for cloud apps

Icon	Command	Description
	(Re)deploy	Deploy or redeploy the selected app.
	Undeploy	Undeploy the selected app.
	Debug	Start debugging the selected application.
	Edit Configuration	Edit the settings for an associated cloud deployment run/debug configuration.

Deployment status icons

Icon	Status
	Unknown
	Deployed
	Undeployed

Command Line Tools Console Tool Window

Tools | Run Command

`Ctrl+Shift+X`

Use the tabs of this tool window to type and run commands in the command line and to validate tool definition files for structure consistence.

The character set to be used in the tool window is chosen from the Console encoding drop-down list on the Command Line Tool Support page of the Settings dialog box.

The tool window consists of the following tabs and areas:

- The Input text box is displayed in the bottom of the tool window if you have chosen Tool window in the Show console in area of the Command Line Tool Support page. Otherwise, the Command Line Tools Input pane opens in a separate pop-up window.
  In this text box, type the desired command in the format `<tool alias> <command>`
- The output tab shows the results of executing commands. The tab is named after the last invoked command.
- The Tool definition file errors tab is not shown by default. The tab is accessible only if any structure discrepancies are detected in the tool definition file.
  Every time you invoke a command, PhpStorm performs full validation of the tool definition file. If the validation fails, PhpStorm displays a Command Line Tool pop-up window with a notification on validation failure. Upon clicking the More link, the Tool definition file errors tab opens showing messages on detected inconsistencies. Each message contains information on the file and the line number where the problem was found, as well as a brief description of the error.

  You can close the tab by clicking the cross on its header. To re-open the tab, again click More in the Command Line Tool notification pop-up window, which remains on the screen until you close it manually.

  Upon validation failure, the tool is marked with the Invalid description icon in the Command Line Tool Support page.

Toolbar Options

Item	Tooltip and Shortcut	Description
	Stop `Ctrl+F2`	Click this button to cancel execution of a command without closing the tool window.
	To previous command `Ctrl+Alt+Up`	Click this button to navigate to the previous command.
	To next command `Ctrl+Alt+Down`	Click this button to navigate to the next command.
	Use Soft Wraps	Click this toggle button to have the soft wrap mode applied to the output.
	Scroll to the end	Click this button to navigate to the bottom of the output tab named after the last invoked command.
	Print	Click this button to have the contents of the console printed out. Upon clicking the button, PhpStorm opens the Print dialog box, where you can configure the printing procedure and output.
	Clear All	Click this button to remove all text from the console. This function is also available on the context menu of the console.
	Export to Text `Alt+O`	Click this button to have the results of executing commands saved in a text file. In the Export Preview dialog box, that opens, specify the target file, and click Save.
	`Ctrl+Shift+F4`	Click this button to close the tool window. If one or more commands are still running, the Command Line Tool dialog box opens. Specify whether you want to stop them or leave running in the background by clicking one of the following buttons: – Terminate and close – Close without terminating
	Help `F1`	Use this icon or shortcut to open the corresponding help page.

Command Line Tools Input Pane

Tools | Run Command

`Ctrl+Shift+X`

In this input pane, type the desired command in the format `<tool alias> <command>`

The input pane is available in two modes:

- As a separate pop-up window, if you have chosen Tool window in the Show console in area of the Command Line Tool Support page.
- As the Input text box at the bottom of the Command Line Tools tool window.

View the results of command execution in the output tab of the dedicated Command Line Tools Console tool window. The tab is named after the last invoked command.

Coverage Tool Window

View | Tool Windows | Coverage

This tool window appears on running PHPUnit tests with coverage or JSTestDriver unit tests with coverage In this tool window, explore the coverage measurement results.

In this section:

- Toolbar
- Context menu

Toolbar

Item	Description
	Click this button to go up one level.
	When this button is pressed, source code of the class selected in the tool window, automatically opens in a separate editor tab, and gains the focus.
	When this button is pressed, when source code of certain class gets the focus in the editor, the corresponding node is automatically highlighted in the tool window.

	Click this button to generate a code coverage report and save it to the specified directory.
	The button is not available when the tests are executed on Karma because a coverage report is actually generated on the disk every time Karma tests are run. The format of a coverage report can be configured in the configuration file, for example:
	
	`// karma.conf.js`
	`module.exports = function(config) {`
	
	`config.set({ ...`
	
	`// optionally, configure the reporter`
	
	`coverageReporter: { type : 'html', dir`
	`: 'coverage/'`
	`}`
	
	`...`
	
	`});};`
	
	The following `type` values are acceptable:
	– `html` produces a bunch of HTML files with annotated source code
	– `lcovonly` produces an `lcov.info` file
	– `lcov` produces HTML + `.lcov` files. This format is applied by default.
	– `cobertura` produces a `cobertura-coverage.xml` file for easy Hudson integration
	– `text-summary` produces a compact text summary of coverage, typically to the console
	– `text` produces a detailed text table with coverage for all files

	Click this button to close the tool window.
	Click this button to show reference.

Context menu

Item	Shortcut	Description
Jump to Source	F4	Choose this command to open the selected file in the editor.

Database Console

From the Database tool window (for any node within a DB data source):

- on the title bar if the toolbar is hidden
- on the toolbar if the toolbar is shown
- Open Console or Open New Console from the context menu
- `Ctrl+Shift+F10`

From the Scratches view of the Project tool window :

- View | Jump to Source
- Jump to Source from the context menu
- `F4`

On this page:

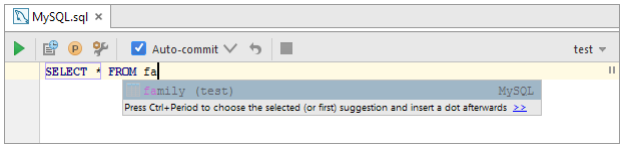
- Overview
- Input pane
- Toolbar of the Database Console tool window
- Output pane
- Result pane
- Parameters pane

See also, Working with Database Consoles .

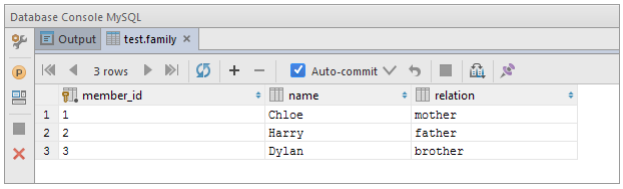
Overview

Database consoles let you compose and execute SQL statements for databases defined in PhpStorm as data sources. They also let you analyze and modify the retrieved data.

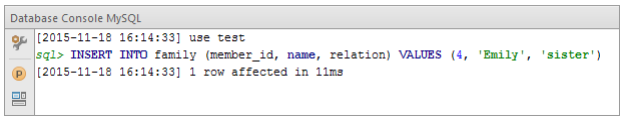
The input pane of a database console opens as a separate editor tab. This is where you compose your SQL statements.



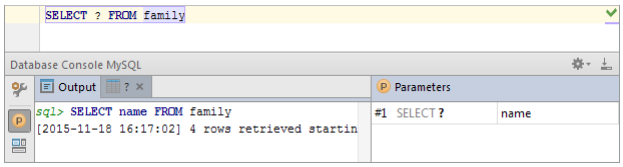
When you execute your first statement ( ), the Database Console tool window opens. If the executed statement retrieves data (e.g. `SELECT`), there are two panes in the tool window shown on the Output and the Result tabs. (The tab showing retrieved data may be labeled Result # or, if appropriate, the table name may be shown.)



Otherwise, only the output pane is shown.



Additionally, you can open the Parameters pane ( ) to manage parameters in SQL statements.



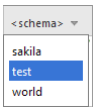
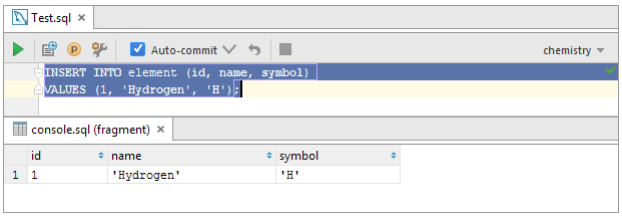
Input pane

Use the input pane to compose and execute your SQL statements as well as to perform other, associated tasks.

The available functions are accessed by means of the toolbar icons, keyboard shortcuts and context menu commands.

- Toolbar icons and shortcuts
- Most useful context menu commands

ItemShortcutDescription		
Toolbar icons and shortcuts		
Execute	`Ctrl+Enter`	Use this icon or shortcut to execute the selected (highlighted) SQL statement or statements. If nothing is selected, the current statement is executed. See also, Run 'console.sql', Execute in Console and Executing an SQL statement.

Browse Console History	Ctrl+Alt+E	Use this icon or shortcut to open a dialog that shows all the statements that you have run for the corresponding data source. See also, Executing auto-memorized statements.
View Parameters		Use this icon to open or close the Parameters pane .
Settings		Use this icon to open the Database page of the Settings dialog to view or edit the settings for your database consoles, the Table Editor and the Database tool window.
Auto-commit		Use this check box to turn the autocommit mode for the database connection on or off. In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back. If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the commit or rollback command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements. In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.
Commit		If the autocommit mode is off, use this icon to commit the current transaction.
Rollback		If the autocommit mode is off, use this icon to roll back the current transaction.
Cancel Running Statements	Ctrl+F2	Use this icon or shortcut to terminate execution of the current statement or statements.
<schema>		A list for selecting the default schema or database (may be unavailable). 
Most useful context menu commands		
Open DML Table Editor		If an `INSERT` statement is currently selected: open the editor for working with the data in table format.  See also, Editing data for INSERT statements in table format.
Change Dialect (<CurrentDialect>)		Use this command to change the SQL dialect being used. Select the necessary dialect from the list. In addition to particular dialects, also the following option is available: – <Generic>. Basic SQL92-based support is provided including completion and highlighting for SQL keywords, and table and column names. Syntax error highlighting is not available. So all the statements in the input pane are always shown as syntactically correct. See also, Changing the SQL dialect.
Explain Plan		Show an execution plan (a.k.a. explain plan) for the current statement. The result is shown in a mixed tree/table format on a dedicated Plan tab.
Explain Plan (Raw)		Show an execution plan (a.k.a. explain plan) for the current statement. The result is shown in table format. (Technically, `EXPLAIN <CURRENT_STATEMENT>` or similar statement is executed.)
Execute	Ctrl+Enter	Execute the current statement or the sequence of selected statements.
Execute to File		Execute the current statement (e.g. `SELECT`) and save the result in a text file. Select the output format, and specify the file location and name.
Run 'console.sql'	Ctrl+Shift+F10	Use this command or shortcut to execute all the statements contained in the console.
Diagrams	Ctrl+Shift+Alt+U Ctrl+Alt+U	If the cursor is within the name of a schema or table: open a UML class diagram for the schema or table. – Show Visualisation (Ctrl+Shift+Alt+U). The diagram opens on a separate editor tab. – Show Visualisation Popup (Ctrl+Alt+U). The diagram opens in a pop-up window.

Toolbar of the Database Console tool window

To hide or show the toolbar, click on the title bar and select Show Toolbar.

Item	Shortcut	Description
Settings		Use this icon to open the Database page of the Settings dialog to view or edit the settings for your database consoles, the Table Editor and the Database tool window.
Enable SYS.DBMS_OUTPUT	Ctrl+F8	For Oracle: use this icon or shortcut to enable or disable showing the contents of the DBMS_OUTPUT buffer in the output pane.
View Parameters		Use this icon to open or close the Parameters pane .
Restore Layout		Use this icon to restore the original tool window layout (after the rearrangements that you have made).
Cancel Running Statements	Ctrl+F2	Use this icon or shortcut to terminate execution of the current statement or statements.
Close	Ctrl+Shift+F4	Use this icon or shortcut to close the tool window.

Output pane

This pane shows the SQL statements that you have run as well as information about other operations performed in the console. These include turning the autocommit mode on or off, committing or rolling back a transaction, etc.

The information about the errors that occur is also shown in this pane.

For most of the events the following information is provided:

- Timestamp, that is, when the event took place.
- For data definition and data manipulation operations - how many rows were affected (e.g. added, changed or deleted). For data retrieval operations - how many rows were retrieved.
- Duration in milliseconds.

Use the following context menu commands:

- Copy (`Ctrl+C`) to copy the text selected in the output pane to the clipboard.
- Compare with Clipboard to compare the text selected in the output pane with the contents of the clipboard.
- Clear All to clear all the contents of the output pane.

Result pane

This pane shows the data retrieved from the database in table format. You can sort, add, edit and remove the data as well as perform other, associated tasks.

- Main functions
- Using the header row

Main functions. Most of the functions in the Result pane are accessed by means of controls on the toolbar, context menu commands for the data cells, and associated keyboard shortcuts.

Item	Shortcut	Description
and		These icons and corresponding commands are for switching between the result set pages, i.e. the pages that show the retrieved data. A fixed number of rows shown simultaneously is referred to as a result set page. If this number is less than the number of rows that satisfy the query, only a subset of all the rows is shown at a time. In such cases, you can use and to switch between the subsets. (If all the rows are currently shown, these icons and the corresponding commands are inactive.) The result set page size is set on the Database page of the Settings dialog.
First Page		Use this icon or command to switch to the first of the result set pages to see the first series of rows.
Previous Page	`Ctrl+Alt+Up`	Use this icon, command or shortcut to switch to the previous result set page to see the previous series of rows.
Next Page	`Ctrl+Alt+Down`	Use this icon, command or shortcut to switch to the next result set page to see the next series of rows.
Last Page		Use this icon or command to switch to the last of the result set pages to see the last series of rows.
Reload Page	`Ctrl+F5`	Use this icon, command or shortcut to refresh the current table view. Use this function to: - Synchronize the data shown with the actual contents of the database. - Apply the Result set page size setting after its change.
Add New Row	`Alt+Insert`	Use this icon, command or shortcut to add a new row to the table. Complete entering a value into a cell by pressing `Enter`. To save the new row, select Submit New Row from the context menu or press `Ctrl+Enter`. If inappropriate in the current context (i.e. for the table currently shown), this function is not available. See also, Adding a row.
Delete Selected Rows	`Ctrl+Y`	Use this icon, command or shortcut to delete the selected row or rows. Rows are selected by clicking the cells in the column where the row numbers are shown. To select more than one row, use mouse clicks in combination with the `Ctrl` key. If inappropriate in the current context (i.e. for the table currently shown), this function is not available.
Auto-commit		Use this check box to turn the autocommit mode for the database connection on or off. In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back. (Each change of a value in the table, or adding or deleting a row results in executing an SQL statement.) If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the commit or rollback command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements. In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.
Commit		If the autocommit mode is off, use this icon or command to commit the current transaction.
Rollback		If the autocommit mode is off, use this icon or command to roll back the current transaction.
Cancel Query	`Ctrl+F2`	Use this icon or shortcut to terminate execution of the current query.
Pin Tab		Use this icon or command to pin the tab to the tool window to keep the query result. See also, Pinning the Result tab.

Tab S...(TSV) Data Extractor		Use this button or command to open a menu in which you can select an output format for your data. The Configure CSV Formats command in that menu opens the CSV Formats Dialog that lets you manage the delimiter-separated values formats (e.g. CSV, TSV).
View Query		Use this button to view the query which was used to generate the table. To close the pane where the query is shown, press Escape.
		This icon provides access to the following commands: – Transpose. Use this command to turn the transposed table view on or off. (In the transposed view, the rows and columns are interchanged. So, the rows are shown as columns and vice versa.) – Reset View. Use this command to restore the initial table view after reordering or hiding the columns, or sorting the data. – Settings. Use this command to open the Database page of the Settings dialog to view or edit the settings for the Database console, the Table Editor and the Database tool window.
Clone Row	Ctrl+D	Use this command or shortcut to create a copy of the selected row.
Edit Value	F2	Use this command or shortcut to start editing a value in the selected table cell. (Alternatively, you can double-click the cell or simply start typing.) To save the modified value, press Enter. To cancel editing, press Escape. See also, Modifying cell contents.
Quick Documentation	Ctrl+Q	Use this command or shortcut to open the quick documentation view. To close the view, press Escape. For more information, see Using the quick documentation view.
Go To | Row	Ctrl+G	Use this command or shortcut to switch to a specified row. In the dialog that opens, specify the row number to go to.
Go To | Related Data	F4	Use this command or shortcut to switch to a related record. The command options are a combination of those for Go To | Referenced Data and Go To | Referencing Data . The command is not available if there are no related records.
Go To | Referenced Data	Ctrl+B	Use this command or shortcut to switch to a record that the current record references. If more than one record is referenced, select the target record in the pop-up that appears. The command is not available if there are no referenced records.
Go To | Referencing Data	Alt+F7	Use this command or shortcut to see the records that reference the current record. In the pop-up that appears there are two categories for the target records: – First Referencing Row. All the rows in the corresponding table will be shown and the first of the rows that references the current row will be selected. – All Referencing Rows. Only the rows that reference the current row will be shown. The command is not available if there are no records that reference the current one.
Dump Data | To Clipboard		Use this command to copy the table data onto the clipboard.
Dump Data | To File		Use this command to save the table data in a file. In the dialog that opens, specify the location and name of the file.
Save LOB		Use this command to save the large object (LOB) currently selected in the table in a file.

Using the header row. In the Result pane, you can use the cells in the header row (i.e. the row where column names are shown) for:

– Sorting data

– Reordering columns

– Hiding and showing columns

You can sort table data by any of the columns by clicking the cells in the header row.

Each cell in this row has a sorting marker in the right-hand part and, initially, a cell may look something like this: name. The sorting marker in this case indicates that the data is not sorted by this column.

If you click the cell once, the data is sorted by the corresponding column in the ascending order. This is indicated by the sorting marker appearance: name 1. The number to the right of the marker (1 on the picture) is the sorting level. (You can sort by more than one column. In such cases, different columns will have different sorting levels.)

When you click the cell for the second time, the data is sorted in the descending order. Here is how the sorting marker indicates this order: name 1.

Finally, when you click the cell for the third time, the initial state is resorted. That is, sorting by the corresponding column is canceled: name.

Here is an example of a table where data are sorted by two of its columns.

	id	name 2	relation 1
1	3	Dylan	brother
2	6	Jack	brother
3	1	Harry	father
4	2	Chloe	mother
5	5	Alice	sister
6	4	Emily	sister

To restore the initial "unsorted" state for the table, click and select Reset View.

To reorder columns, use drag-and-drop for the corresponding cells in the header row. To restore the initial order of columns, click and select Reset View.

	member_id	relation	name 1
1	5	sister	lice
2	1	mother	hloe
3	3	brother	ylan
4	4	sister	mily
5	2	father	arry

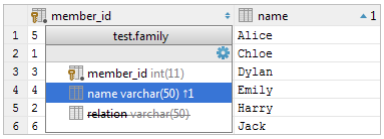
To hide a column, right-click the corresponding header cell and select Hide column.

To show a hidden column:

Do one of the following:

1. Do one of the following:
   - Right-click any of the cells in the header row and select Column List.
   - Press `Ctrl+F12`.

   In the list that appears, the names of hidden columns are shown struck through.



2. Select (highlight) the column name of interest and press `Space`.
3. Press `Enter` or `Escape` to close the list.

To show all the columns, click and select Reset View.

See also, Using the Structure view to sort data, and hide and show columns.

Parameters pane

The Parameters pane shows the parameters detected in the input pane and lets you edit their values. To open or close this pane, use on the toolbar.

To start editing a value, switch to the corresponding table cell and start typing. To indicate that you have finished editing a value, press `Enter` or switch to a different cell.
To quit the editing mode and restore an initial value, press `Escape`.

When you select a row in the table, the corresponding parameter is highlighted in the input pane.

See also, Executing parameterized statements.

Database Tool Window

View | Tool Windows | Database

On this page:

- Overview of the tool window
- Toolbar icons, context menu commands and shortcuts
- View options
- Icons for data sources and their elements

See also, Working with the Database Tool Window .

Overview of the tool window

The Database tool window provides access to functions for working with databases and DDL data sources . It lets you view and modify data structures in your databases, and perform other associated tasks.

The available data sources are shown as a tree of data sources, schemas, tables and columns. If no data sources are currently defined, use the New command (`Alt+Insert`) to create a data source.

Most of the functions in this window are accessed by means of the toolbar icons or context menu commands. (If the toolbar is not currently shown, click on the title bar and select Show Toolbar.) Many of the commands have keyboard shortcuts. If the toolbar is hidden, the Synchronize and Open Console commands can be access by means of the title bar icons (and respectively).

on the title bar provides access to view options.

Toolbar icons, context menu commands and shortcuts

Icon	Command	Shortcut	Description	Available for
			Use this icon on the title bar to collapse all the nodes.	All node types
	New	`Alt+Insert`	Use this icon, command or shortcut to create a new data source, database console, table, column, index, or a primary or foreign key. The list of options depends on which element is currently selected. See also, Creating a data source, Data Sources and Drivers Dialog , Creating and opening a new database console, and Creating a table, a column, an index, or a primary or foreign key.	DB data sources and their elements. If a DDL data source is selected, you can only choose to create another data source.
	Duplicate	`Ctrl+D`	Use this icon, command or shortcut to create a copy of the selected data source. Specify the properties of the data source in the Data Sources and Drivers dialog that opens. See Data Sources and Drivers Dialog .	DB and DDL data source nodes
	Synchronize	`Ctrl+Alt+Y`	Use this icon, command or shortcut to update the view of the selected element (i.e. to synchronize the view of the element with its actual state in the database). See also, Auto sync .	DB data sources and their elements
	Properties		Use this icon or command to open the Data Sources and Drivers dialog to manage your data sources and their settings.	All node types
	Disconnect	`Ctrl+F2`	Use this icon, command or shortcut to close the database connection for the selected DB data source or data sources. (The names of the data sources with active database connections are shown in bold.)	DB data sources with active connections and their elements
	Table Editor	`F4`	Use this icon, command or shortcut to open the selected database table in the Table Editor . See also, Working with the Table Editor .	Tables and table columns in DB data sources
	View Editor, Routine Editor, Package Editor	`F4`	Use this icon, command or shortcut to open the definition of the selected view, function, procedure or package in the editor.	Corresponding elements in DB data sources
	Edit Source	`F4`	Use this icon, command or shortcut to open the associated DDL file in the editor.	Tables and table columns in DDL data sources
	Open Console	`Ctrl+Shift+F10`	Use this icon, command or shortcut to open the default database console for the corresponding DB data source.	DB data sources and their elements (tables and table columns)
	Drop Associated		Use the Drop Associated | Primary Key command to remove primary key constraints.	Tables and columns in DB data sources
	Rename	`Shift+F6`	Use this command or shortcut to rename the selected data source, table or table column. Specify the new name in the dialog that opens. See also, Renaming a table or column.	All node types
	Modify Table, Modify Column,	`Ctrl+F6`	Use this command or shortcut to edit the definition of the selected table, column, index or foreign key. See also, Modifying the definition of a table, column, index or a	Corresponding elements in DB data sources

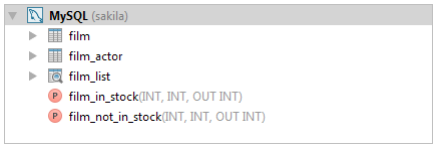
	Modify Index, Modify Foreign Key		foreign key.	
	Copy Reference	`Ctrl+Shift+Alt+C`	Use this command or shortcut to copy the fully qualified name of the selected data source, table or table column to the clipboard.	All node types
	Drop or Remove	`Delete`	Use this command or shortcut to remove the selected item.	All node types
	Find Usages	`Alt+F7`	Use this command or shortcut to find the usages of (references to) the selected item (data source, table or table column) in the project source files and libraries.	All node types
	Compare	`Ctrl+D`	Select two data sources, schemas or tables and then use this command or shortcut to compare table structures for the selected items. The comparison results are shown in the differences viewer .	DB and DDL data sources and tables
	Open New Console		Use this command to create and open a new database console for the corresponding DB data source.	DB data sources and their elements (tables and table columns)
	Forget Schemas		Use this command in problematic cases such as when your data structures start to display incorrectly, fail to synchronize, etc. As a result, PhpStorm deletes the information it has accumulated about your database. To check if this has eliminated the problem, use the Synchronize command .	DB data sources
	Copy DDL	`Ctrl+Shift+C`	Use this command or shortcut to copy a DDL definition of the selected data source, table or column onto the clipboard.	All node types
	Dump Data to File		Use this command to save the data for the selected table or view in a file, or to view or edit the settings for delimiter-separated values output formats. Select the output format or Configure CSV Formats. In the latter case, the CSV Formats Dialog will open and you'll be able to edit the settings for existing formats, and define new output formats.	Tables in DB data sources
	Color Settings		Use this command if you want to set or change the color for the selected element or elements. (The Database Color Settings dialog will open.)	All node types

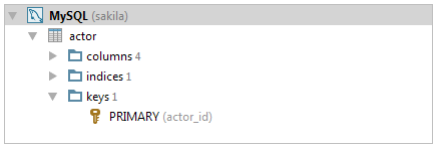
	Diagrams	`Ctrl+Shift+Alt+U`
		
		`Ctrl+Alt+U`
			Use this command or shortcut to see a UML class diagram for the selected data source or table. Select:
			– Show Visualisation to open the diagram on a separate editor tab.
			– Show Visualisation Popup to see the diagram in a pop-up window.
				DB and DDL data sources and tables

		`Ctrl+Q`	Use this key combination to see basic information about the selected element. For example, the info about a table includes the names of the data source, catalog, schema and the table itself, the table definition (`CREATE TABLE`) and, if appropriate, the first 10 rows. To close the documentation pop-up, press `Escape` . You can also access this function from the main menu: View | Quick Documentation.	All node types

View options

The view options, generally, define what is shown in the tool window and how. To view or change these options, click ⚙ on the title bar.

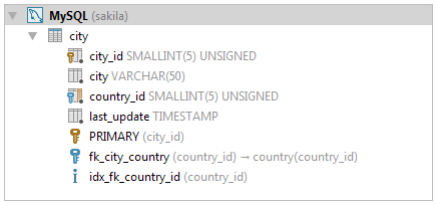
Option	Description
Group Schema	This option defines how schema elements are shown. When on, there are separate nodes for tables, views and stored routines (shown as folders). Tables, views and routines (procedures and functions) are shown as elements of the corresponding groups. When off, there is no explicit grouping for tables, views, and routines. Tables and views are followed by procedures and functions. 
Group Contents	This option defines how table elements are shown. When on, there are separate nodes for columns, indexes, primary and foreign key constraints, and triggers (shown as folders). The elements appear in the corresponding groups.



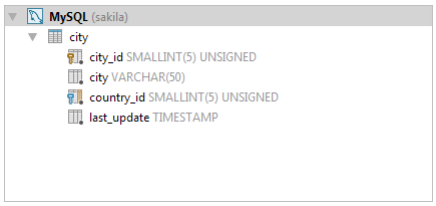
When off, there is no such grouping and, generally, only columns are shown for tables.

Show Keys and etc.

When this option is on, the public and foreign key constraints, and indexes are shown as separate elements.



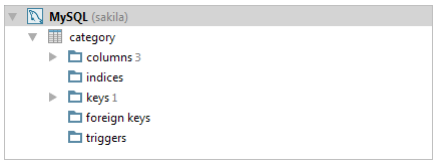
Otherwise, there are no separate elements for the keys and indexes.



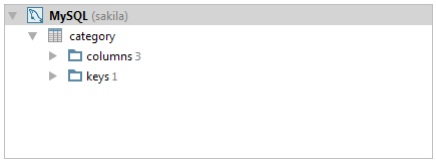
The option is unavailable when the Group Contents option is on.

Show Empty Groups

If the Group Schema or the Group Contents option is on, you can select to show or hide empty groups, i.e. the categories that contain no elements.
The Show Empty Groups option is on:

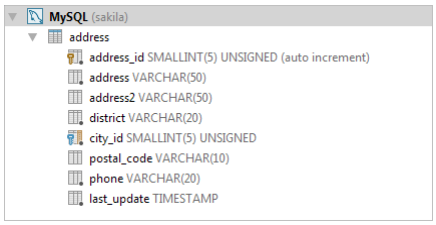


The Show Empty Groups option is off:

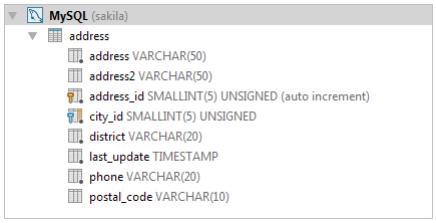


Sort Alphabetically

When this option is off, columns, generally, are unsorted.



When this option is on, the columns are ordered alphabetically.



Show Toolbar

Select or deselect this option to show or hide the toolbar.

The rest of the options are common for all the tool windows, see Viewing Modes .

Icons for data sources and their elements

Icon	Description
	DB data source. Also, DBMS-specific icons are used: DB2

	Derby H2 HSQLDB MySQL Oracle PostgreSQL SQL Server SQLite Sybase
	DB data source with the read-only status, e.g. for DB2.
	DDL data source
	Schema
	Table
	View
	Column
	A `NOT NULL` column
	Column with a primary key
	Column with a foreign key
	Column with an index
	Primary key
	Foreign key
	Index
	Stored procedure or function

Data Sources and Drivers Dialog

To access this dialog from the Database tool window : or + on the toolbar

Use this dialog to manage your data sources and database drivers, and their settings.

A driver here is understood as a collection that includes database driver files, and also default options and settings for creating a DB data source.

The names of new, yet unsaved items are shown in the left-hand pane in green. New items are saved when clicking Apply or OK.

- Left-hand pane
- DB data source settings
- DDL data source settings
- Driver settings
- Problems

See also, Managing Data Sources .

Left-hand pane

Shown in this pane are your data sources and drivers . When you select an item, its settings are shown in the right-hand part of the dialog.

Use the toolbar icons, context menu commands and associated keyboard shortcuts to manage your data sources and drivers, and also to perform other, related tasks.

Icon and command	Shortcut	Description
+ Add	Alt+Insert	Use this icon, command or shortcut to create a new data source or driver. Select: – The name of DBMS to create a DB data source associated with the corresponding DBMS. If you have created your own drivers, you can also select a driver to use it as the basis for creating the data source. See DB data source settings . – DDL Data Source to create a DDL data source. See DDL data source settings . – Database Driver to create a driver. See Driver settings .
— Remove	Delete	Use this icon, command or shortcut to remove the selected item or items from the list.
Duplicate	Ctrl+D	Use this icon, command or shortcut to create a copy of the selected data source or driver.
Go to Driver	Ctrl+B	Use this icon, command or shortcut to view or edit the settings for the driver associated with the selected DB data source.
Make Global or Move to Project		Use this icon or command to move the selected DB data source to the global or project level. (Global data sources are available in all your projects.)
Change Driver		If more than one driver is available for the selected DB data source, use this command to associate the data source with a different driver.
Show Driver Usages	Alt+F7	Use this command or shortcut to find the DB data sources that use the selected driver. The found data sources are shown in the Used By popup which lets you navigate to anyone of the found data sources.
Reset Changes	Ctrl+Alt+Z	Use this command or shortcut to undo the changes made to the selected item.
		Use these icons to switch between the items you've been workings with.

DB data source settings

- Name
- General tab
- SSH/SSL tab
- Schemas tab
- Options tab for Oracle
- Advanced tab

Name

Item	Description
Name	Use this field to edit the name of the data source.

General tab

Shown on this tab are mainly the database connection settings.

The user interface is adjustable: the set of available controls depends on which option is selected in the list to the right of the URL field .

Item	Description
File	If your database is a local file, specify the path to that file. ... lets you select an existing database file. + lets you create a new database file.
Path	If your database is a local file or folder, specify the path to that file or folder. ... lets you select the database file or folder. Create database. Select this option to create a new database. (This option may be unavailable.)

Host	Specify the hostname (domain name) or the IP address of the computer on which the database is located. If the database is on your local computer, specify `localhost` or `127.0.0.1` . If you are using SSH, the database host must be accessible by the specified domain name or IP address from the computer on which the SSH proxy runs. See SSH/SSL tab .
Port	Specify the database port number.
Database	Specify the name of the target database or schema.
User	Specify the name of the database user (i.e. your database user account name).
Password	Specify the password for the database user .
Save on disk with master password protection	Select this check box if you want PhpStorm to save the password in its password database. Later, instead of the password, you'll be asked for your master password. (The master password is used to access the password database.)
URL	Shown in this field is the URL that PhpStorm will use to connect to the database. The user interface for specifying the URL is different depending on which option is selected in the list to the right: – URL only. This option, generally, is for editing the database connection URL directly. When you select this option, only the following fields are available: User , Password and URL. You should edit the URL right in the field. Your user name and password, if necessary, are specified in the corresponding fields, or within the URL in the format appropriate for the JDBC driver that you are using. – When using any other of the options (the options are DBMS-specific), PhpStorm forms the database connection URL automatically using the info in the fields above the URL field. In all such cases, normally, you don't need to edit the URL (though you can if you want).
Test Connection	Click this button to make sure that the database connection settings are correct and PhpStorm can communicate with the target database.
Driver	Click the <driver name> link to switch to the settings for the associated driver.
Read-only	Select this check box if you want to protect the data source from accidental data modifications. As a result, you won't be able to modify the data in the Table Editor . Whether data modifications will be possible by means of the consoles depends on the DBMS: PhpStorm will try to set the database connection status to read-only. All the rest depends on the database driver, i.e. whether and to which extent the driver supports the read-only status. For example, if you define your MySQL data source as read-only, the driver won't let you switch schemas in associated database consoles (see e.g. Selecting the default schema or database). In that case, you should use the Database field to specify your current schema or database.
Auto commit	The default setting for the Auto-commit option in the table editor and the database console .
Auto sync	If this option is off, the view of the data source in the Database tool window is synchronized with the actual state of the database only when you perform the Synchronize command (or `Ctrl+Alt+Y`). If this option is on, the view of the data source is automatically updated: – When you change the data source settings. (Technically, when you click OK in the Data Sources and Drivers dialog.) – When you run DDL SQL statements in the database consoles associated with the data source. Note that auto sync is performed for the overall database and, thus, may be time-consuming. So auto-synchronization is more suitable for "small" databases. If your database is "big", it's recommended that you sync its state manually (e.g. `Ctrl+Alt+Y`), and only for the appropriate database parts such as separate tables.

SSH/SSL tab

If the target database should be accessed using SSH or SSL, select the corresponding check box and specify the associated settings.

– SSH

– SSL

Item	Description
SSH	
Use SSH tunnel	Select this check box to set up and use an SSH tunnel for accessing a remote database via an SSH proxy.
Copy from	If there is already a data source for which the necessary SSH settings are specified, you can copy those settings from that data source. Click the link and select the data source to copy the settings from.
Proxy host	Specify the hostname (domain name) or IP address of the SSH proxy server that you are using. The SSH proxy server host must be accessible by the specified hostname or IP address from your local computer.
Port	Specify the port on which your SSH proxy server accepts SSH connections. The port number `22` suggested by PhpStorm is the standard port used by SSH servers. Change this number if your SSH proxy server uses a different port.
Proxy user	Specify the name of the SSH proxy user.
Auth type	Specify the user authentication type used by your SSH proxy. Select: – Password for password-based authentication. – Key pair (Open SSH) for key-based authentication.
Proxy password	For password-based authentication: specify the password for the SSH proxy user. See also, Save on disk .
Private key file	For key-based authentication: specify the path to the file where the corresponding private key is stored. Type the path in the field, or click (`Shift+Enter`) and select the file in the dialog that opens .
Passphrase	For key-based authentication: specify the passphrase for the private key if the key is locked with the passphrase.
Save on disk with master password protection	Select this check box if you want PhpStorm to save the password or the passphrase in its password database. Later, instead of the password or the passphrase, you'll be asked for your master password. (The master password is used to access the password database.)

SSL. All the files specified in this section should be in PEM format. Which of the files you have to specify, depends on the SSL properties of your user account.

Use SSL	Select the check box to use SSL when connecting to the server.
Copy from	If there is already a data source for which the necessary SSL settings are specified, you can copy those settings from that data source. Click the link and select the data source to copy the settings from.
CA file	Specify the path to SSL Certificate Authority (CA) certificate file. If used, this must be the same certificate as used by the server. Type the path in the field, or click ... (`Shift+Enter`) and select the file in the dialog that opens .
Client certificate file	Specify the path to your (client) public key certificate file. Type the path in the field, or click ... (`Shift+Enter`) and select the file in the dialog that opens .
Client key file	Specify the path to your (client) private key file. Type the path in the field, or click ... (`Shift+Enter`) and select the file in the dialog that opens .

Schemas tab

Specify which schemas and database objects (tables, views, etc.) in the target database you are going to be working with.

You might need to wait a little while PhpStorm is retrieving the necessary structure information from the database.

Item	Description
Schemas	Shown in this column are the schemas found in the database. Use the check boxes to the left of schema names to select the schemas you want to be working with. Only the selected schemas will be shown in the Database tool window.
Resolve unqualified references	To be able to reference the tables by their "short" names (e.g. in the database console or in your SQL files), select the corresponding check boxes. Otherwise, only fully qualified table names will be treated by PhpStorm as valid. Note that the DBMS may reject short names when you execute the corresponding statement or statements.
Object filter	To further limit the set of tables and other database objects shown in the Database tool window, specify the corresponding filter. The specified filter is applied to "short" names of the database objects in the selected schemas .
Use legacy introspector	If this check box is present, an older version of the database introspector is available for the corresponding DBMS. (An introcpector is a component that analyzes the database structure.) If there are problems with retrieving the database structure information for your database, you may want to select the check box to use the older introspector version.

Options tab for Oracle

This tab lets you specify which table should be used to store the EXPLAIN PLAN output information.

Item	Description
Explain plan	Selecting None disables the Explain Plan command.
Plan table	The name of the `EXPLAIN PLAN` output table.

Advanced tab

On this tab, you can configure the database connection properties, and also specify the options and environment variables for the database driver JVM.

Item	Description
Name - Value	The set of connection options passed to the database driver as key - value pairs at its start. When you select a cell in the Name column, the description of the corresponding option is shown underneath the table. To find an option of interest, just start typing its name. To start editing a value, click or double-click the corresponding Value field, or press `F2` . To add a row, start editing the values in the last row, where <user defined> and <value> are shown. A new row will be added to the table automatically.
VM Options	The options for the JVM in which the database driver runs. (The driver is started as a separate process in its own JVM.) Example. For certain Oracle Database versions (e.g. for version 9), there may be connection problems when you and your database server are in different time zones. Specifying the time offset for your timezone may help, e.g. `-Duser.timezone=UTC+03:00` . Alternatively, you can try setting the variable `oracle.jdbc.timezoneAsRegion` to `false` in the Name - Value table .
VM Environment	Environment variables for the database driver JVM. Example. Sometimes, when working with Oracle, your data and/or error messages don't display correctly. Many of such problems are encoding-related and can be solved by appropriately setting the `NLS_LANG` variable, e.g. `NLS_LANG=Russian_CIS.CL8MSWIN1251` . For more information, see e.g. Oracle NLS_LANG FAQ . To start editing the variables, click ... .

For additional information, refer to your DBMS documentation.

DDL data source settings

A DDL data source is defined by its name, and can reference one or more DDL files and/or another data source (a parent data source).

Item	Description
Name	Use this field to edit the data source name.
DDL Files	Use the controls in this area to compose the list of files that contain the necessary DDL definitions. + (`Alt+Insert`). Use this icon or shortcut to add a DDL SQL file or files to the data source definition. In the dialog that opens, select the necessary file or files.

- (Alt+Delete). Use this icon or shortcut to remove the selected file or files from the list.
- (Alt+Up). Use this icon or shortcut to move the selected file one line up in the list.
- (Alt+Down). Use this icon or shortcut to move the selected file one line down in the list.

Extend	If necessary, select another data source as a parent. As a result, the data source whose properties you are editing will "inherit" all the DDL definitions from its parent. If the parent data source is not needed, select <none>.

Driver settings

- Name
- Settings tab
- Advanced tab

Name

Item	Description
Name	Use this field to edit the name of the driver .

Settings tab

Shown on this tab are mainly the defaults for the General tab .

Item	Description
Class	The fully qualified name of the driver class to be used.
Dialect	The SQL dialect associated with the corresponding data sources. Via the data sources this dialect will "propagate" to the database console . In addition to particular dialects, also the following option is available: - <Generic>. Basic SQL92-based support is provided including completion and highlighting for SQL keywords, and table and column names. Syntax error highlighting is not available. So all the statements in the input pane are always shown as syntactically correct.
Auto commit	The default setting for the auto commit option .
Auto sync	The default setting for the auto sync option .
URL templates	The templates used to construct the database URL. The text in curly brackets represents variables, e.g. - `{host}` the domain name or IP address of the database host. - `{port}` the database port number. - `{database}` the name of the database or schema. Optional fragments are in square brackets, e.g. `[:{port}]` . Template names correspond to the names of the options in the URL options list .
Driver files	Shown in this area is the list of the driver files. Use: - The download link to download the necessary driver files. - (Alt+Insert) to add the files available locally to the list. - (Alt+Delete) to remove unnecessary files from the list. (No files are physically deleted.)

Advanced tab

Shown on this tab are the default settings for the Advanced tab .

Item	Description
Name - Value	The default set of connection options passed to the database driver as key - value pairs at its start. To start editing a value, double-click the corresponding Value field. To add a row, start editing the values in the last row, where <user defined> and <value> are shown. A new row will be added to the table automatically.
VM Options	The default options for the JVM in which the database driver runs. (The driver is started as a separate process in its own JVM.)

Problems

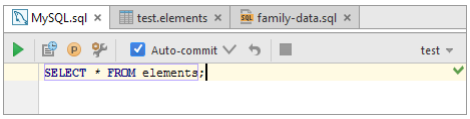
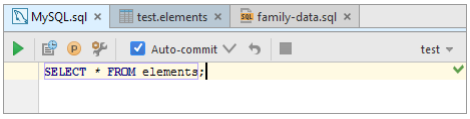
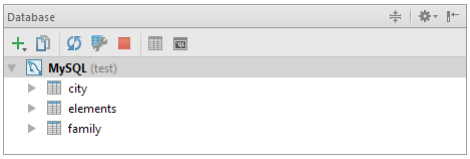
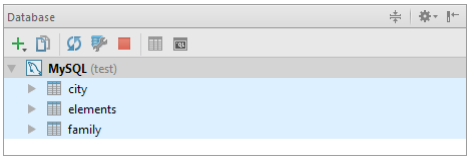
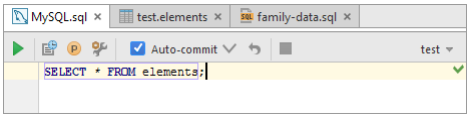
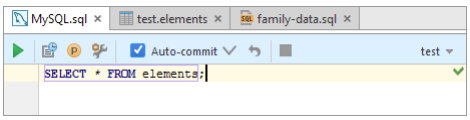
If potential problems are detected, there is a number to the right of Problems. In that case, if you click Problems, you'll see the list of problems as well as controls for fixing them.

Database Color Settings Dialog

From the Database tool window :

Select the element or elements of interest, and select Color Settings from the context menu.

Select the color , and specify how this color should be used (see Shared and Set recursively). Use the check boxes in the Appearance Settings section to enable or disable the use of the database colors in various places in the UI.

Item	Description
Color	Click the color that you want to use for the element or elements selected in the Database tool window. If the color that you want is missing, click Custom and specify the color in the dialog that opens. The color which is currently selected is marked with its name, e.g. Blue. If you don't want to use any color, click No Color .
Shared	Select the check box if you want to share the change of color you are about to make with your team. (The database color settings are shared through version control.) In technical terms, the new color (for the selected element or elements) will be stored in – `.idea/databaseColors.xml` or the `.ipr` file if the check box is selected. – `.idea/workspace.xml` or the `.iws` file if the check box is not selected. The `workspace.xml` or the `.iws` file, normally, is not shared through version control while the rest of PhpStorm configuration files are. For more information on PhpStorm configuration files and their sharing, see Project .
Set recursively	If the check box is selected, the selected color will be applied to the element itself and all its subordinate elements recursively. If the check box is not selected, the colors set individually for the subordinate elements won't change.
Appearance Settings	
Enable database colors	Clear the check box to (temporarily) disable the use of the database colors everywhere in the UI.
Use in editor tabs	Select the check box to enable the use of the database colors for editor tabs (when working with the input pane of the database console and the table editor). When this option is off, the tabs look like this:  When this option is on, the tabs have the same color as the corresponding data source or table (e.g. blue): 
Use in database view	Select the check box to enable the use of the database colors in the Database tool window. When this option is off, the tool window looks like this:  When this option is on, the tool window may look as shown below. (In this example, the blue color is set for the corresponding data source.) 
Use in toolbars	Select the check box to use the database colors to paint the background of the toolbars in the input pane of the database console and the table editor. When this option is off, the toolbar looks like this:  When this option is on, the toolbar background has the same color as the corresponding data source (e.g. blue). 

No Color	Click this button, if you don't want to use any color for the selected element or elements.

Debug Tool Window

View | Tool Windows | Debug

`Alt+5`

In this topic:

- Overview
- Debug toolbar
- Stepping toolbar
- Hide/restore toolbar
- Moving tabs and areas
- Context menu of a tab

Overview

The Debug tool window becomes available when you start debugging .

It displays the output generated by the debugging session for your application. If you are debugging multiple applications, the output for each application is displayed in a separate tab named after the corresponding run/debug configuration .

For each application, there are the following nested tabs:

- Console : displays system information and error messages, and the console input and output of your application.
- Debugger tab divided into the following areas:
  - Frames
  - Variables
  - Watches
- Elements : appears if you are using Chrome browser for debugging.

Each area has a context menu that allows you to configure its behavior and navigate between tabs.

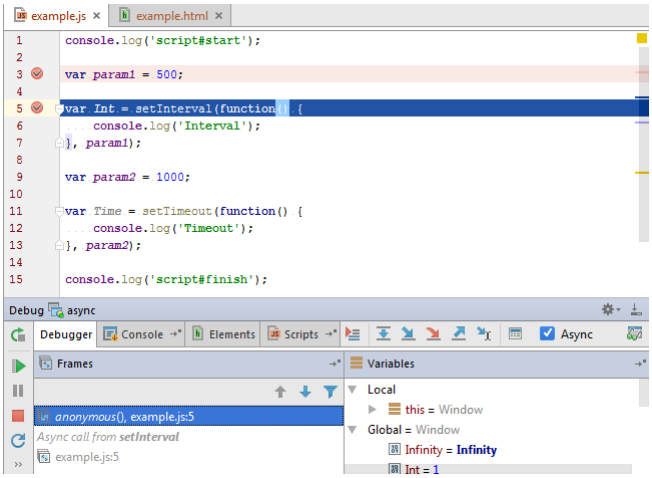
Each of the tabs and areas can be hidden/restored , or moved to a location of your choice.

Debug toolbar

Item	Tooltip and Shortcut	Description
	Rerun `Ctrl+F5`	Click this button to stop the current application and run it again. When an application is stopped, this button toggles to .
	Debug `F9`	When the current application is stopped, click this button to debug it again. When an application is running, this button toggles to .
	Resume Program `F9`	When an application is paused, click this button to resume program execution.
	Pause Program `Ctrl+Pause`	Click this button to pause program execution. Note that the button is not available for Run/Debug Configuration: Node JS , Run/Debug Configuration: Node JS Remote Debug , and Run/Debug Configuration: NodeUnit .
	Stop `Ctrl+F2`	Click this button to terminate the current process externally by means of the standard `shutdown` script. For a Node.js run configuration , clicking the button once invokes soft kill allowing the Node.js application to catch the `SIGINT` event and perform graceful termination (on Windows, the `Ctrl+C` event is emulated). After the button is clicked once, it is replaced with indicating that subsequent click will lead to force termination of the application, e.g. on Unix `SIGKILL` is sent.
	View Breakpoints `Ctrl+Shift+F8`	Click this button to open the Breakpoints dialog box where you can configure breakpoints behavior.
	Mute Breakpoints	Use this button to toggle breakpoints status. When the button is pressed in the toolbar of the Debug tool window, all the breakpoints in a project are muted, and their icons become grey: . You can temporarily mute all the breakpoints in a project to execute the program without stopping at breakpoints.
	Restore Layout	Click this button abandon changes to the current layout and return to the default state.
	Settings	Click this button to open the menu with the following options available: – Show Values Inline: select this option to enable the Inline Debugging feature that allows viewing the values of variables right next to their usage in the editor. – Sort Values Alphabetically: select this option to sort the values in the Variables pane in the alphabetical order. – Unmute breakpoints on session finish: select this option to re-enable all disabled breakpoints after the debugging session has been finished.
	Pin	Click this button to pin or unpin the currently selected tab.
	Close `Ctrl+Shift+F4`	Click this button to close the selected tab.

	Help F1	Click this button to open the corresponding help page.

Stepping toolbar

ItemTooltip and Shortcut	Description
Show Execution Point Alt+F10	Click this button to highlight the current execution point in the editor and show the corresponding stack frame in the Frames pane.
Step Over F8	Click this button to execute the program until the next line in the current method or file, skipping the methods referenced at the current execution point (if any). If the current line is the last one in the method, execution steps to the line executed right after this method.
Step Into F7	Click this button to have the debugger step into the method called at the current execution point.
Force Step Into Shift+Alt+F7	Click this button to have the debugger step into the method called in the current execution point even if this method is to be skipped.
Step Out Shift+F8	Click this button to have the debugger step out of the current method, to the line executed right after it.
Run to Cursor Alt+F9	Click this button to resume program execution and pause until the execution point reaches the line at the current cursor location in the editor. No breakpoint is required. Actually, there is a temporary breakpoint set for the current line at the caret, which is removed once program execution is paused. Thus, if the caret is positioned at the line which has already been executed, the program will be just resumed for further execution, because there is no way to roll back to previous breakpoints. This action is especially useful when you have stepped deep into the methods sequence and need to step out of several methods at once. If there are breakpoints set for the lines that should be executed before bringing you to the specified line, the debugger will pause at the first encountered breakpoint. Tip Use this action when you need a kind of a temporary breakpoint at a specific line, where program execution should not be interrupted.
Evaluate Expression Alt+F8	Click this button to open the Evaluate Expression dialog.
Async	– When this check box is selected, PhpStorm recognizes breakpoints inside asynchronous code and stops at them and lets you step into asynchronous code. As soon as a breakpoint inside an asynchronous function is hit or you step into asynchronous code, a new element `Async call from <caller>` is added in the Frames pane of the Debugger tab. PhpStorm displays a full call stack, including the caller and the entire way to the beginning of the asynchronous actions. – When the check box is cleared, PhpStorm does not recognize and therefore skips breakpoints in the asynchronous code and does not allow you to step into it. The image below shows an example of a JavaScript debugging session.  – With the Async check box selected, the debugger will stop at line3(breakpont), then at line5(breakpoint). On clicking Step into, the debugger will stop at line5 (on `function`), then will move to line6. – With the Async check box cleared, the debugger will stop at line3(breakpont), then at line5(breakpoint). On clicking Step into, the debugger will move to line9.

Hide/restore toolbar

IconTooltip	Description
Hide	Click this button located in the upper-right corner of the Debug Console, Watches, Treads, Frames, or Variables pane to hide the corresponding area. When an area is hidden, its icon appears in upper-right corner of the Debugger.
Restore 'Console' view	Click this button to make the Console area visible. This button becomes available after clicking .

	Restore 'Frames' view	Click this button to make the Frames area visible. This button becomes available after clicking .
	Restore 'Watches' view	Click this button to make the Watches area visible. This button becomes available after clicking .
	Restore 'Variables' view	Click this button to make the Variables area visible. This button becomes available after clicking .

Moving tabs and areas

If you are unhappy with the default layout of the Debug tool window, you can always move the tabs and areas. To to that, just drag a tab or an area to the desired location. The possible target gets highlighted:

Drop the tab or area in the highlighted location.

To restore the default layout of tabs and area, click in the Debug toolbar.

Context menu of a tab

Use the context menu of the Frames / Threads , Variables or Watches areas to configure the behavior of these areas or navigate between tabs.

Item	Description
Hide	Click this button to hide the corresponding area
Close Others	Click this button to hide all tabs except for the Console and Debugger tabs.
Focus On Startup	If this option is selected, the selected area gets the focus when you start a debugging session.
Focus On Breakpoint	If this option is selected, the selected area gets the focus when a breakpoint is reached.
Select Next Tab / Select Previous Tab	Use these options to switch between the Console and the Debugger tabs.

`Ctrl+Alt+Right` / `Ctrl+Alt+Left`

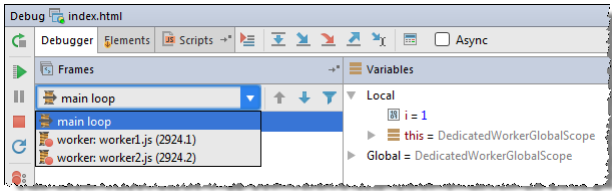
Debug Tool Window. Debugger

The Debugger tab is divided into the following areas:

- Frames
- Variables
- Watches

Debug Tool Window. Frames

The Frames pane shows the execution path that led to the point where code was paused. When you are debugging Web/Service Workers or during debugging a multi-thread Node.js application after an additional Node.js process starts, see Debugging Workers and Running and Debugging Node.js respectively, the drop-down list on the top of the pane becomes available. Use the drop-down list to jump between the Workers threads and the main application thread or between threads of a multi-thread Node.js application.



To examine a thread, select it from the drop-down list on top of the pane. The status and type of a thread is indicated by a special icon and a textual note next to the thread's name.

To examine the values stored in a frame, use the Variables pane of the Debug tool window.

Toolbar

Item	Shortcut and Tooltip	Description
	Previous Frame/Next Frame `Ctrl+Alt+Up` / `Ctrl+Alt+Down`	Use the arrow buttons to navigate through the frame stack.
	Hide Frames from Libraries	Click this button to hide frames from libraries. If this button is released, all frames are displayed.

Debug Tool Window. Variables

In this topic:

- Overview
- Toolbar
- Context menu
- Variable types

Overview

The Variables pane enables you to examine the values stored in the objects of your application.

When a stack frame is selected in the Frames pane , the Variables pane displays all data within its scope (method parameters, local and instance variables).

In this pane, you can set labels for objects, inspect objects, evaluate expressions, add variables to watches and more.

Toolbar

This toolbar appears only on clicking the button in the Watches pane . So doing, the focus is with the Debugger tab .

Item	Shortcut	Description
+	Alt+Insert	Click this button to create a new watch.
−	Alt+Delete	Click this button to remove the selected watch from the list.
	Alt+Up Alt+Down	Use these buttons to change the order of watches.
	Ctrl+D	Use this button to create a copy of the selected watch.
	N/A	Use this button to show watches in the Variables pane . When this button is pressed, the corresponding Variables pane shows this toolbar and the created watches.

Context menu

Item	Shortcut	Description
Inspect	N/A	This command is available for fields, local variables and reference expressions, and opens a non-modal Inspection window, where you can concentrate on a particular reference. You can open as many Inspection windows as required. The view in the Inspection window is the same as in the Watches pane, but requires less screen space.
Mark Object	F11	Use this command to add an object label.
Set Value	F2	Use this command to change the runtime value of a field or a variable.
Copy Value	Ctrl+C	Use this command to copy the value of the selected variable to the Clipboard. If multiple items are selected, not only variables' values, but also their structure is copied, so that when you copy-paste the selection to a text file, the indentation mimics the tree output of the debugger to produce an easy-to-read output. Alternatively, hover your mouse cursor over a value and view its contents in the tooltip.
Copy Value As	N/A	This menu item is available only in the PHP context. Choose this command to copy the selected variable to the Clipboard in one of the following formats: − print_r − var_export − json_encode
Copy JSON		This menu item is available only in the JavaScript context. Choose this command to copy the selected value in the JSON format .
Compare Value with Clipboard	N/A	Use this command to compare the selected value with the value currently in the Clipboard.
Copy Name	N/A	Use this command to copy the name of the selected variable to the Clipboard.
Evaluate Expression	Alt+F8	Use this command to evaluate the selected variable in the dialog that opens.
Evaluate In Console	N/A	The option is available in the JavaScript context only. Choose this option to have the selected variable copied as input in the Console pane. When you switch to the Console pane, the variable name is shown in green at > and its value is displayed below in blue. Alternatively you can type the name of any variable at > in the Console manually and press Enter to have its value displayed. Code completion is at your disposal: as you type the name of a variable, PhpStorm displays a suggestion list. Note that this functionality is available only when the Use Console Input toggle button is pressed. To evaluate a previously evaluated variable without searching for it in the Variables pane, find the variable in question using the Up and Down arrows on you keyboard and press Enter .
Add to Watches	N/A	This command is available for all nodes except static ones. Use this command to create an expression that references the node and add this expression to the Watches pane.
Show Referring Objects	N/A	Use this command to display a list of objects referring to the currently selected variable.
Jump to Source	F4	This command opens the source code of the selected variable or field in the editor and places the caret in the corresponding line.
Jump to Type Source	Shift+F4	Use this command to open the source code of the selected variable or field type in the editor.

Variable types

The icon on the left of each variable indicates its type:

- : static
- : global
- : field

- : array
- : primitive
- : object

Debug Tool Window. Watches

In this topic:

- Overview
- Toolbar
- Context menu

Overview

In the Watches pane you can evaluate any number of variables or expressions in the context of the current stack frame. The values are updated with each step through the application, and become visible every time the application is suspended.

While the Evaluate Expression command on the context menu of the Variables pane enables you to see one expression at a time, the Watches pane shows multiple expressions that persist from one debug session to another, until you remove them.

You can create watches in this pane, in the Variables pane and even in the editor.

Watch expressions are always evaluated in the context of a stack frame that is currently inspected in the Frames pane. If an expression cannot be evaluated, it is displayed with a question mark.

Toolbar

Item	Shortcut	Description
+	Alt+Insert	Click this button to create a new watch.
−	Alt+Delete	Click this button to remove the selected watch from the list.
↑↓	Alt+Up Alt+Down	Use these buttons to change the order of watches.
	Ctrl+D	Use this button to create a copy of the selected watch.
	N/A	Use this button to show watches in the Variables pane . When this button is pressed, the corresponding Variables pane shows this toolbar and the created watches.

Context menu

Item	Shortcut	Description
New Watch	Insert	Choose this command to create a new watch. A text field opens, where you can enter new watch expression.
Remove Watch	Delete	Choose this command to delete the currently selected watch expression from the list.
Edit	F2	Choose this command to change the selected watch expression.
Remove All Watches		Choose this command to delete all watch expressions from the list.
Inspect		Available for fields, local variables and reference expressions. Choose this command to open the Inspect window for the node, which allows you to perform the same operations as those available in the stack frame, with the only difference that the root node is the one you have selected. You can recursively call the new Inspect windows from within each other. Each window is not modal and immediately reflects all changes in its subtree.
Jump to Source	F4	This command opens the source code of the selected variable or field in the editor and places the caret on a proper line.
Copy Value	Ctrl+C	Copies the selected node value to the clipboard.
Copy Name		Copies the selected node name to the clipboard.

Debug Tool Window. Console

This tab is marked with and shows the output and error stream messages. In the PHP and JavaScript context, you can view the values of variables right in the Debug tool window during a debugging session.

On this page:

- Console Toolbar
- Context Menu Options
- Keyboard Shortcuts
- Evaluating Variables in the Console Pane
- Executing Code in the Console Pane during a Debugging Session

Console Toolbar

Item	Tooltip and shortcut	Description
	Up/down the Stack Trace Ctrl+Alt+Up Ctrl+Alt+Down	Click this button to navigate up or down in the stack trace and have the cursor jump to the corresponding location in the source code.
	Use Soft Wraps	Click this button to toggle the soft wrap mode of the output.
	Scroll to the end	Click this button to navigate to the bottom of the stack trace and have the cursor jump to the corresponding location in the source code.
	Print	Click this button to send the console text to the default printer.
	Clear All	Click this button to remove all text from the console. This function is also available on the context menu of the console.
	Use Console Input	– When this toggle-button is pressed, you can evaluate and alter variables, call PHP functions, and define additional functions on the fly in the Console pane. – When the toggle-button is released, any input in the Console pane will be treated like `STDIN`.

Context Menu Options

Item	Description
Compare with Clipboard	Opens the Clipboard vs Editor dialog box that allows you to view the differences between the selection from the editor and the current clipboard content. This dialog is a regular comparing tool that enables you to copy the line at caret to the clipboard, find text, navigate between differences and manage white spaces.
Copy URL	Choose this command to copy the current URL to the system clipboard. This command only shows on a URL, if it is included in an application's output.
Create Gist	Choose this command to open the Create Gist dialog box.
Clear All	Clears the output window.

Keyboard Shortcuts

The Ctrl+D key combination allows you to send EOF (end of file), i.e. to signal that no more data can be read from a data source.

Evaluating Variables in the Console Pane

In the PHP and JavaScript context, you can view the values of variables right in the Debug tool window during a debugging session. This can be done in several ways:

- Select the variable in question in the Variables pane and choose Evaluate in Console on the context menu of the selection. When you switch to the Console pane, the variable name is shown in green at `>` and its value is displayed below in blue.
- Alternatively you can type the name of any variable at `>` in the Console manually and press Enter to have its value displayed. Code completion is at your disposal: as you type the name of a variable, PhpStorm displays a suggestion list. Note that this functionality is available only when the Use Console Input toggle button is pressed.
- To evaluate a previously evaluated variable without searching for it in the Variables pane, find the variable in question using the Up and Down arrows on you keyboard and press Enter.

Executing Code in the Console Pane during a Debugging Session

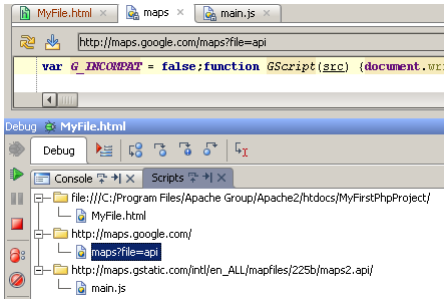
During a PHP debugging session, you can not only evaluate variables, but also change their values, call PHP functions, and define additional ones right in the Console pane. Note that this functionality is not available for PHPUnit or Behat run/debug configurations.

1. Start a debugging session (see PHP Debugging Session) and switch to the Console pane of the Debugger tab.
2. Press the Use Console Input toggle button on the toolbar.
3. Type a statement or expression at `>` and press Enter. PhpStorm evaluates your code fragment and shows the output below the input code. You can type most PHP constructs including class declarations, function declarations, variables, expressions, etc. Code completion is at your disposal: as you type, PhpStorm displays a suggestion list.
4. When typing a multi-line code fragment, press Shift+Enter to start a new line and Ctrl+Enter to split a line.
5. Output for arrays and objects is by default wrapped in a `var_export()` function and displayed in the Console. To hide the output displayed, clear the Show array and object children in Debug Console check box on the Debug page.

Debug Tool Window: Scripts

Warning! The tab is only available during a JavaScript debugging session.

The tab shows all the external resources that are automatically downloaded during the JavaScript code execution. Use this tab to monitor downloading the external resources. To open the code of a resource in the editor, double-click the item in question.



Debug Tool Window: Elements Tab

In this tab, view the HTML source code that implements the active browser page and its HTML DOM structure . Any changes made to the page through the browser are immediately reflected in the tab.

To Have the Tab Displayed:

1. Make sure you are using the Chrome browser for debugging.
2. Make sure the JetBrains Chrome Extension is installed in your Chrome browser.
   The extension is installed when you first initiate a debugging session and follow the dedicated link provided by PhpStorm. For details, see Using JetBrains Chrome Extension .
3. Make sure the LiveEdit repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

The Structure, Text, and Scripts Panes

The tab consists of three panes: Structure, Text, and Scripts.

The Structure pane shows the HTML DOM structure of the page that is currently active in the browser. The structure is updated dynamically according to the changes made on the page.

The Text pane shows HTML source code of the page that is currently active in th browser. The code is updated dynamically according to the changes made on the page.

The Scripts pane shows a tree of executed scripts.

The Structure and Text panes are mutually synchronized. When you click a node in the DOM structure, PhpStorm scrolls through the contents of the Text pane.

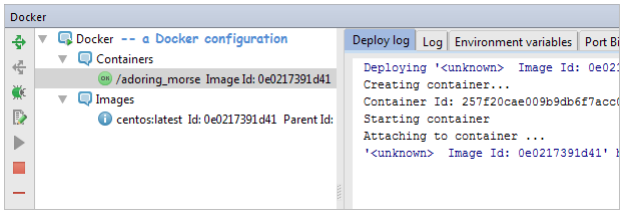
The panes are also synchronized with the browser. PhpStorm highlights the element in the browser as soon as you click the corresponding node in the DOM structure or in the Text pane.

Docker tool window

View | Tool Windows | Docker

For the tool window to be available, the Docker integration plugin must be installed and at least one Docker configuration must be defined.

The Docker tool window lets you manage your Docker images and containers.



All the available functions are accessed by means of the toolbar icons and context menu commands.

See also, Docker .

Icon	Command	Description
Docker configurations		
	Connect	Connect to Docker Remote API. As a result, the list of available Docker images and containers is shown.
	Disconnect	Disconnect from Docker Remote API. As a result, the list of Docker images and containers is hidden.
	Edit Configuration	Edit the Docker Remote API settings for the selected configuration.
	Deploy	Deploy an image or artifact using the deployment settings in one of the Docker Deployment run configurations. The run configuration is selected from the list that is shown.
	Debug	Start one of the Docker Deployment run configurations in the debug mode. The run configuration is selected from the list that is shown.
	Pull image	Pull an image from Docker Hub or other image repository, e.g. Quay . (The Pull Image dialog will open.)
Docker images		
	Create container	Create a container for the selected image according to an existing or new Docker Deployment run configuration.
	Delete image	Delete the selected image or images.
	Push image	Push the selected image to Docker Hub or other image repository, e.g. Quay . (The Push Image dialog will open.)
	Copy image ID	Copy an ID of the selected image to the clipboard.
Docker containers		
	(Re)deploy	Deploy an image or artifact using the deployment settings in the corresponding Docker Deployment run configuration.
	Debug	Start an associated run configuration in the debug mode.
	Edit Configuration	Edit the settings for an associated run configuration.
	Start container	Start the selected container.
	Stop container	Stop the selected container.
	Delete container	Delete the selected container or containers.
	Open browser	If a web app is deployed in the container: open the default web browser.
	Copy container ID	Copy an ID of the selected container to the clipboard.
	Copy image ID	Copy an ID of the associated image to the clipboard.
	Show log	Show the container log.
	Inspect	Show low-level container information in JSON format.
	Show processes	Show the list of processes running in the container.
	Attach	Open the console for working with standard streams (stdin, stdout and stderr).
	Exec	Run the `docker exec` command. For `docker exec [OPTIONS] CONTAINER COMMAND [ARG...]` , you should specify only the `COMMAND [ARG...]` part. See exec .

Documentation Tool Window

View | Tool Windows | Documentation

`Ctrl+Q`

This tool window appears in the following cases:

- Quick Documentation lookup is pinned by clicking in the upper-right side of the lookup window.
- Quick Definition tooltip is pinned by clicking in the upper-right corner of the tooltip.

As a result, the tool window appears in the list of available tool windows in the View menu, and gets the corresponding sidebar icon Documentation .

Icon	Shortcut	Description
	`Left` or `Right`	Switch to the previous or next documentation page (e.g. after using hyperlinks). **Note** On an OS X computer, you can also use the three-finger right-to-left and left-to-right swipe gestures.
	`Shift+F1`	View external documentation in the default browser.
	`F4`	Switch to the item (e.g. source) that corresponds to the documentation page currently shown.
		Turn the Auto-update from source option on or off. When the option is on, the information in the tool window is synchronized with your navigation in the editor and other places in the UI.
		Click this icon to show the font size slider. Move the slider to increase or decrease the font size as required.

Duplicates Tool Window

View | Tool Windows | Duplicates

The Duplicates tool window displays results of the search for duplicates.

On this page:

- Panes of the Duplicates tool window
- Left toolbar
- Upper toolbar
- Context menu commands

Panes of the Duplicates tool window

The window consists of the following panes:

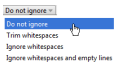
- The left pane displays the tree view of the duplicate fragments of source code. Each node shows the following information:
  - The number of duplicated code fragments found in scope.
  - The 'cost' of the duplicate (which is an arbitrary unit calculated using an additive algorithm on the base of the code block size; generally, the larger is the code fragment, the higher is its cost).
  - The containing class where the duplicates are located.
- The right pane shows the differences between the duplicated fragments of source code, selected in the left pane.

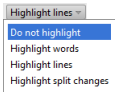
Left toolbar

Item	ShortcutDescription	
	Rerun	Click this button to rerun the duplicates analysis in the active tab.
	Close Active Tab `Ctrl+Shift+F4`	Click this button to close the active tab.
	Autoscroll to Source	If the button is pressed, selecting an entry in the left pane opens the respective file in the editor.
	`F1`	Click this button to show reference.

Upper toolbar

ItemTooltip/Image/	ShortcutDescription	
	`F7` / `Shift+F7`	Move to the nxt/previous difference

Whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

Highlighting mode		Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted – Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Jump to Source `F4`	Click this button to open the file in the active pane in the editor. The caret will be placed in the same position as in the Duplicates tool window.
	Synchronize scrolling	Click this button to simultaneously scroll both differences panes; if this button is released, each of the panes can be scrolled independently.
	Editor settings	Click this button to reveal the drop-down list of available check commands. Select or clear these check commands to show or hide line numbers, indentation guides, white spaces, and soft wraps. Note that same commands are duplicated in the context menu of the differences viewer gutter. Right-click the gutter to show the menu of check commands.

Context menu commands

ItemKeyboard Shortcut	Description	
Jump to Source	`F4`	Open in the editor the file that contains the selected duplicate, and place the caret at the beginning of the duplicate. The fragment of

		code is highlighted.
Show Source	`Ctrl+Enter`	Open in the editor the file that contains the selected duplicate, and highlight the fragment of code.
Send to left/Send to right		Use these commands, or the arrow icons , to place the selected duplicate to the left or right pane of the differences viewer.

Event Log

View | Tool Windows | Event Log

Event Log in the right-hand part of the bottom tool window bar

---

The Event Log tool window shows the information about "important" events that take place in PhpStorm.

The information about problematic situations (e.g. errors and exceptions) is displayed in red. In such cases, clicking the more link (if present) opens a balloon with a more detailed description of the error or exception. Clicking a description link (depending on the error, the text may be different e.g. NullPointerException) opens the IDE Fatal Errors dialog which lets you review the error and create a bug report.

- General tab
- Database tab
- Toolbar

General tab

The General tab appears when the Database tab opens. In the absence of the Database tab, the information is shown in the output pane which, in this case, is not tabbed.

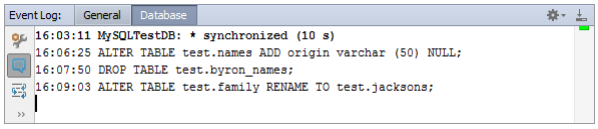
Database tab

Shown on the Database tab are the events related to working with the Database tool window and the Table Editor .

For the Database tool window, the following may be shown:

- The information about data synchronizations and various manipulations with databases such as creating, modifying and deleting tables and columns, etc.
- Error messages.

For the Table Editor only error messages are shown.



Toolbar

Item	Description
Settings	Use this icon to open the Notifications page where you can select which events you want to be notified of and also which events should be logged.
Show balloons	Use this icon to enable or disable showing notifications. Note that this icon is a toggle which turns the corresponding feature on or off. When showing notifications is enabled, you are notified of the events that take place in PhpStorm. The corresponding notifications are shown in the balloons. The alternative way to enable or disable this feature is by means of the Display balloon notifications check box on the Notifications page .
Use Soft Wraps	Use this icon to turn on or off the soft wrap mode for the output.
Scroll to the end	Use this icon to go to the end of the event log.
Mark all as read	Use this icon to mark all the messages as read.
Clear all	Use this icon to delete all the messages and thus clear the log. This function can alternatively be accessed by means of the context menu.
Help (F1)	Use this icon or shortcut to open the corresponding help topic.

Favorites Tool Window

View | Tool Windows | Favorites

Alt+2

The Favorites tool window lets you manage the following lists:

- Favorite project items . Each list of Favorites in the tool window appears with the star icon.
- Bookmarks . The list of bookmarks is marked with icon.
- Breakpoints . The list of breakpoints is marked with icon.

Initially, the lists are empty. So doing, the favorites list has the same name as the project.

The lists of bookmarks and breakpoints are filled in automatically, as the new bookmarks or breakpoints are added.

To add items to favorites, do one of the following:

- Select one or more items in the Project tool window or the Find tool window . Then choose File | Add To Favorites.
- Right-click an editor tab, and choose Add to Favorites or Add All to Favorites.

On this page:

- Toolbar buttons
- Context menu commands
- Title bar context menu and buttons
- Context menu of the side bar button
- Using drag-and-drop

Toolbar buttons

Item	Shortcut	Description
+	Alt+Insert	Use this button to create a new list of favorite items. In the Add New Favorites List dialog, specify the name for the new list and click OK.
	Ctrl+Enter	Click this button to rename the selected list.
−	Alt+Delete	Use this button to delete the selected list or list item. Depending on the selection, the following behaviors take place: – If a Favorites list, or one of the nested favorites is selected, click this button to delete an entire favorites list, or the selected item. So doing an item or a list is deleted from the Favorites tool window, but is left intact in project. – If a bookmark under the Bookmarks list is selected, click this button to delete this bookmark from the list and from the source code. Vice versa, if a bookmark is deleted from the source code, it is also removed from the Favorites tool window. – If a breakpoint under the Breakpoints list is selected, click this button to delete the selected breakpoint from the list and from the source code. Vice versa, if a breakpoint is deleted from the source code, it is also removed from the Favorites tool window.

Context menu commands

When you right-click an item in the content pane, the context menu for this item is shown. This menu provides access to all the functions available for the selected item.

The following table lists and explains the commands for managing the favorites lists and their contents. Other commands, functionally, are similar to those in the Project tool window .

Item	Shortcut	Description
Rename Favorites List		Use this command to change the name of the selected list. In the New Name For Favorites List dialog, edit the name of the list and click OK.
Remove From Favorites	N/A	Use this command to delete the selected list or list item. Note that there may be list items corresponding to certain structures (for example, folders with subfolders and files inside). These may be deleted only as a whole, that is, you cannot delete their individual structural elements. To differentiate between the list items and their structural elements (if any), you can use the path information as a visual hint. Such information is shown for favorites list items but not for their elements.
Delete Favorites List '<list_name>'		Use this command to delete the list with the name `<list_name>` . As a rule, this is the list selected in the tool window.
Delete All User-Defined Lists Except '<list_name>'		Use this command to delete all user-defined favorites lists except the one with the name `<list_name>` . (The list with the name `<list_name>` , as a rule, is the one currently selected in the tool window.)
Add to Favorites		Use this command to add (i.e. copy) the selected list item to a different list. To add the item to an existing favorites list, select the target favorites list. To add the item to the list that does not yet exist, select Add To New Favorites List. In the Add New Favorites List dialog, specify the name of the new list and click OK. As a result, a new list is created and the selected item is added to this list.
Send To Favorites		Use this command to move the selected list item to a different list. To move the item to an existing favorites list, select the target favorites list. To move the item to the list that does not yet exist, select Send To New Favorites List. In the Add New Favorites List dialog, specify the name

of the new list and click OK. As a result, a new list is created and the selected item is moved to this list.

Title bar context menu and buttons

The title bar context menu provides the options for controlling the tool window viewing modes . It also contains the commands for associating the tool window with a different tool window bar , resizing and hiding the tool window.

To access the menu, right-click the window title bar.

Note that most of the menu options may alternatively be accessed by means of the title bar buttons.

Toolbar icon	Context menu command	Description
	Pinned, Docked, Floating, Split Mode	These options let you control general appearance and behavior of the tool window, see Viewing Modes .
	Move to	To associate the tool window with a different tool window bar , select this command, and then select the destination tool window bar (Top, Left, Bottom or Right).
	Resize	To resize the tool window by moving one of its borders, select this command, and then select the necessary Stretch to option. Note that this command is not available in the floating mode.
Shift+Escape	Hide	Use this command to hide the tool window. Tip When used in combination with the Alt key, clicking this button hides all the tool windows attached to the same tool window bar .
		Use this button to collapse all expanded nodes.

Context menu of the side bar button

Right-click the side bar button to reveal this context menu. Refer to the section Viewing Modes for the detailed information about the viewing modes.

Item	Description
	Use this command to turn the Show Members option on or off. If this option is on, the class members (fields, methods, etc.) are shown.
	Use this command to turn the Autoscroll to Source option on or off. If this option is on, PhpStorm automatically navigates from a file (or a class member) selected in the Favorites tool window to the corresponding source file (or its fragment) in the editor. If the corresponding file is not currently open, it will open automatically.
	Use this command to turn the Autoscroll from Source option on or off. If this option is on, PhpStorm automatically navigates from a file (or a class member) selected in the editor, to the corresponding file in the Favorites tool window.

Using drag-and-drop

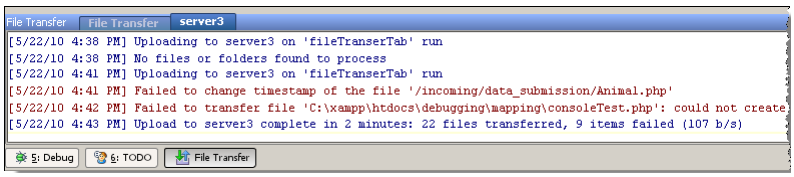
You can use drag-and-drop to:

- Add items to favorites: drag the item of interest from the Project tool window and drop it onto the desired favorites list in the Favorites tool window.
- Drag an external item from the Explorer/Finder and drop it onto the desired favorites list in the Favorites tool window.
- Move an item from one favorites list to another in the Favorites tool window.

File Transfer Tool Window

View | Tool Windows | File Transfer

Use this tool window to view logs while uploading files or folders to a remote server via the FTP or SFTP protocol.



Find Tool Window

View | Tool Windows | Find

`Alt+3`

On this page:

- Basics
- Toolbar buttons
- Context menu commands

Basics

Find tool window displays results of the following searches:

- Find/Replace in Path
- Find Usages
- Refactoring Preview
- Find Usages of a data source , a table, or a column.

The results of each search are displayed in a separate tab, or replace the contents of the current tab, depending on the Open in new tab dialog setting. By default the window appears at the bottom of the screen.

It has a toolbar with a set of buttons, a pane of results, and additional buttons for Replace in Path , and Refactoring Preview operations.

Warning! Unless you find something, this tool window is not visible in the View menu.

Toolbar buttons

Item	Tooltip and shortcut	Description
	Options `Ctrl+Shift+Alt+F7`	Click this button to open one of the Find Usages dialogs, which corresponds to the symbol in question. You can edit the search settings and click Rerun button to execute the modified search query.
	Rerun `Ctrl+F5`	Rerun the last search. This button is not available for viewing code coverage results .
	Close `Ctrl+Shift+F4`	Close the current tab or the tool window. This button is not available in Replace in Path and Refactoring Preview dialogs.
	Pin	Use to pin or unpin the tab. If a tab is pinned, the results for the next command are shown on a new tab.
	Recent find usages `Ctrl+E`	Show the list of recent searches. Select an item in the list to see the search results.

	Expand all
	`Ctrl+NumPad Plus`
	Collapse all
	`Ctrl+NumPad -`
		Use these buttons to have all nodes expanded or collapsed.

	Previous/next occurrence
	`Ctrl+Alt+Up`
	`Ctrl+Alt+Down`
		Navigate to the previous/next element in the tab of results.

	Autoscroll to source	Turns the Autoscroll to source option on or off. When the option is on and you select the search result, the corresponding source file opens in the editor and the appropriate fragment is highlighted in the file.
	Favorites	Click this button to add found usages to favorites .
	Export to Text File `Alt+O`	Save the contents of the current result tab. In the Export preview dialog, specify the target file or copy information to the clipboard. Before saving, you can also modify the information to be saved.
	Help `F1`	Use this icon or shortcut to open the corresponding help page.
	Group by usage type `Ctrl+T`	This button is available for Find Usages only. If this button is pressed, the search results are grouped by the following categories: - comments - unclassified usage not related to any of the categories
	Group by test/production	If this button is pressed, the usages are grouped according to the Production and Test scopes.
	Group by module	If this button is pressed, the found usages show under the

	Ctrl+D	corresponding module or library node. Tip This type of grouping is helpful, when a package is split between several modules.
	Group by package Ctrl+P	If this button is pressed, all the usages found are displayed under their respective packages.
	Group by file structure Ctrl+M	If this toggle is on, the found usages are shown under the corresponding method nodes.
	Merge usages from the same line Ctrl+F	If this toggle is on, the duplicate usages found on the same line are merged.
	Show read access Ctrl+R	This button is available for Find Usages only. If this button is pressed, the search results include references to the read access methods.
	Show write access Ctrl+W	This button is available for Find Usages only. If this button is pressed, the search results include references to the write access methods.
	Show import statements Ctrl+I	This button is available for Find Usages only. If this button is pressed, the search results include the usages in the import statements.
	Preview usages	Turns showing the Preview pane on or off.
	Sort members alphabetically	Click this button to have all members sorted alphabetically. Otherwise, members are sorted in the order they are declared.

Context menu commands

Item	Shortcut	Description
Jump to Source	F4	Navigate to the selected item in the source code.
Include	Insert	For an excluded item : include the item in the list of results.
Exclude	Delete	Exclude the selected item from the list of results. (Excluded items are shown strikethrough.) When you carry out the Replace All or the Do Refactor command, the excluded items are not affected.
Remove	Alt+Delete	Remove the selected item from the list of results.
Recent Find Usages	Ctrl+E	Show the list of recent searches. Select an item in the list to see the search results.
Add to Favorites		Add selected node to the Favorites list .

Framework Tool Window

View | Tool Windows | Framework

Warning! The tool window is available only in a Symfony2 or a Yii specific projects with the MVC view enabled .

In this tool window, examine the project in the terms of bundles, controllers, and other elements of the MVC pattern.

Grunt Tool Window

Context menu of a Gruntfile.js - Show Grunt Tasks

View | Tool Windows | Grunt

On this page:

- Accessing the Grunt Tool Window
- Building a Tree of Grunt Tasks
- Running Grunt Tasks and Targets
- Toolbar
- Context Menu of a Tree
- Context Menu of a Task or a Target

Accessing the Grunt Tool Window

View | Tool Windows | Grunt

- the tool window can be accessed this way only after you have opened it using the Show Grunt Tasks command.

The tool window is available only when:

1. The Node.js runtime environment is installed on your computer.
2. The NodeJS repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. The `grunt-cli` package is installed globally and the `grunt` package is installed in the current project, see Installing Grunt for details.
4. At least one `Gruntfile.js` file is available in the current project.

The tool window opens when you invoke Grunt by choosing Show Grunt Tasks on the context menu of a `Gruntfile.js` in the Project tool window or of a `Gruntfile.js` opened in the editor.

As soon as you invoke Grunt, it starts building a tree of tasks according to the `Gruntfile.js` on which it was invoked. If a task has targets , the task is displayed as a node and the targets are listed under it.

If you have several `Gruntfile.js` files in your project, you can build a separate tasks tree for each of them and run tasks without abandoning the previously built tasks trees. Each tree is shown in a separate tab.

Building a Tree of Grunt Tasks

To build a tasks tree, do one of the following:

- Select the required `Gruntfile.js` file in the Project tool window and choose Show Grunt Tasks on the context menu of the selection.
- Open the required `Gruntfile.js` file in the editor and choose Show Grunt Tasks on the context menu of the editor.
- If the Grunt tool window is already opened, click on the toolbar and choose the required `Gruntfile.js` file from the list. PhpStorm adds a new node and builds a tasks tree under it. The title of the node shows the path to the `Gruntfile.js` file according to which the tree is built.

Running Grunt Tasks and Targets

To run a task or a target from the tree, do one of the following:

- Double click the required task or target in the tree.
- Select the required task or target and choose Run <task name> on the context menu of the selection.
- Select the required task or target and press Enter.
- To run the `default` task, select the root node in the tree and choose Run default on the context menu of the selection.
- To navigate to the definition of a task or target, select the required task or target in the tree and choose Jump to source on the context menu of the selection.

The task or target execution output will be displayed in the Run tool window . The name of the target is shown in the format `<task name>:<target name>` . The tool window shows the Grunt output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed task is displayed on the title bar of the tool window.

To run several tasks or targets, use the multiselect mode: hold Shift (for adjacent items) or Ctrl (for non-adjacent items) keys and select the required tasks or targets, then choose Run on the context menu of the selection.

Toolbar

Item	Tooltip	Description
	Add Gruntfile	Click this button to have a tasks tree for another `Gruntfile.js` file built. Choose the required `Gruntfile.js` file from the pop-up list. PhpStorm adds a new node and builds a tasks tree under it.
	Remove Gruntfile	Click this button to remove the tasks tree under the selected node.
	Reload tasks	Click this button to have the tasks tree under the selected node re-built. You may need a tree re-built after updating the corresponding `Gruntfile.js` file because Grunt does not apply changes to trees on the fly.

Context Menu of a Tree

Item	Description
Grunt Settings	Choose this menu item to open the Grunt Settings dialog box and view or edit the Node.js configuration
Jump to Source	Choose this menu item to open the `Gruntfile.js` file for which the current tree is built.

Reload tasks	Choose this menu item to have the tree of tasks under the selected node re-built.
Copy Path	Choose this menu item to save the path to the `Gruntfile.js` file according to which the current tree was built to the clipboard.
Remove Gruntfile.js	Choose this menu item to remove the tree of tasks under the selected node.

Context Menu of a Task or a Target

Item	Description
Run <task/target name>	Choose this menu item to run the selected task or target.
Debug <task/target name>	Choose this menu item to debug the selected task or target.
Edit <task/target name> settings	Choose this menu item to open the Run/Debug Configuration dialog box and edit the predefined settings for the selected task or target.
Jump to Source	Choose this menu item to open the `Gruntfile.js` file for which the current tree is built and navigate to the definition of the selected task or target.

Gulp Tool Window

On this page:

- Accessing the Gulp Tool Window
- Building a Tree of Gulp Tasks
- Running Gulp Tasks
- Toolbar
- Context Menu of a Tree
- Context Menu of a Task or a Target

Accessing the Gulp Tool Window

Context menu of a Gulpfile.js - Show Gulp Tasks

View | Tool Windows | Gulp - the tool window can be accessed this way only after you have opened it using the Show Gulp Tasks command or after you have run tasks through the Gulp.js run configuration.

The tool window is available only when:

1. The Node.js runtime environment is installed on your computer.
2. The NodeJS repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. The `gulp` package is installed in the current project, see Installing Gulp for details.
4. At least one `Gulpfile.js` file is available in the current project.

The tool window opens when you invoke Gulp.js by choosing Show Gulp Tasks on the context menu of a `Gulpfile.js` in the Project tool window or of a `Gulpfile.js` opened in the editor. The tree is built according to the `Gulpfile.js` file on which Gulp.js was invoked. If you have several `Gulpfile.js` files in your project, you can build a separate tasks tree for each of them and run tasks without abandoning the previously built tasks trees. Each tree is shown under a separate node.

If you have several `Gulpfile.js` files in your project, you can build a separate tasks tree for each of them and run tasks without abandoning the previously built tasks trees. Each tree is shown under a separate node.

Building a Tree of Gulp Tasks

To build a tree of tasks, do one of the following:

- Select the required `Gulpfile.js` file in the Project tool window and choose Show Gulp tasks on the context menu of the selection.
- Open the required `Gulpfile.js` file in the editor and choose Show Gulp tasks on the context menu of the editor.
- If the Gulp tool window is already opened, click on the toolbar and choose the required `Gulpfile.js` file from the list. PhpStorm adds a new node and builds a tasks tree under it. The title of the node shows the path to the `Gulpfile.js` file according to which the tree is built.

Running Gulp Tasks

To run a task, do one of the following:

- Double click the required task in the tree.
- Select the required task and choose Run <task name> on the context menu of the selection.
- Select the required task and press `Enter`.
- To run the `default` task, select the root node in the tree and choose Run default on the context menu of the selection.
- To navigate to the definition of a task, select the required task in the tree and choose Jump to source on the context menu of the selection.

The task execution output will be displayed in the Run tool window .

To run several tasks, use the multiselect mode: hold `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys and select the required tasks, then choose Run on the context menu of the selection.

Toolbar

Item	Tooltip	Description
	Add Gulpfile	Click this button to have a tasks tree for another `Gulpfile.js` file built. Choose the required `Gulpfile.js` file from the pop-up list. PhpStorm builds a tasks tree and shows it under a separate node.
	Remove Gulpfile	Click this button to remove the tasks tree under the selected node.
	Reload tasks	Click this button to have the tasks tree under the selected node re-built. You may need a tree re-built after updating the corresponding `Gulpfile.js` file because Gulp.js does not apply changes to trees on the fly.

Context Menu of a Tree

Item	Description
Gulp Settings	Choose this menu item to open the Gulp Settings dialog box and view or edit the Node.js configuration
Jump to Source	Choose this menu item to open the `Gulpfile.js` file for which the current tree is built.
Reload tasks	Choose this menu item to have the tree of tasks under the selected node re-built.
Copy Path	Choose this menu item to save the path to the `Gulpfile.js` file according to which the current tree was built to the clipboard.
Remove Gulpfile.js	Choose this menu item to remove the tree of tasks under the selected node.

Context Menu of a Task or a Target

Item	Description

Run <task name>	Choose this menu item to run the selected task.
Debug <task name>	Choose this menu item to debug the selected task.
Edit <task name> settings	Choose this menu item to open the Run/Debug Configuration dialog box and edit the predefined settings for the selected task.
Jump to Source	Choose this menu item to open the `Gulpfile.js` file for which the current tree is built and navigate to the definition of the selected task.

Hierarchy Tool Window

View | Tool Windows | Hierarchy

Alt+8

Use this tool window to analyze and navigate through hierarchies of classes .

Warning! The tool window is available only after you have built a hierarchy for the first time.

The contents of the tool window are not automatically updated as you navigate through the source code or switch between the editor tabs. The tool window shows the results of the latest hierarchy command and is updated when you run the next hierarchy command, unless the tab with one of the previously built hierarchies is pinned .

Toolbar buttons

Item	Description	Available In
	When this button is pressed, the hierarchical tree shows both the parent and child classes of the selected class, which is marked with an arrow in the tree of results.	Class hierarchies
	Depending on the hierarchy type: – For class hierarchies - when this button is pressed, the tool window shows all classes that extend the selected class. – For call hierarchies - when this button is pressed, the tool window shows the callees of the selected method.	Class hierarchies Call hierarchies
	Depending on the hierarchy type: – For class hierarchies - when this button is pressed, the tool window shows the hierarchy of each supertype of the current class. – For call hierarchies - when this button is pressed, the tool window shows the callers of the selected method.	Class hierarchies Call hierarchies
	When this button is pressed, the elements within a tree are sorted alphabetically.	All hierarchies
Scope	Use this drop-down list to limit the scope of the current hierarchy: – Project - PhpStorm traces usages of the method across the project. – Test - PhpStorm traces usages of the method across the test classes. – All - PhpStorm traces usages of the method across the project and the libraries. – This class - the scope is limited to the current class. In addition to the pre-configured scopes, you can define your own one: select Configure from the drop-down list, and define the required scope in the Scopes dialog box that opens.	Call hierarchies

In a method hierarchy, the tree-views of the following classes are available:

– + - where this method is defined.
– — - where this method is not defined, but defined in the superclass.
– ! - where this method should be defined, because the class is not abstract.

	If you made any changes of the classes or the class structure, they become visible in the Hierarchy tool window only after you press this button.	All hierarchies
	Toggle the Autoscroll to source mode. When the button is pressed, every time the node is focused, the corresponding line of source code is highlighted in the editor.	All hierarchies
	When this button is pressed, the current tab will not be overwritten; instead, the results of the next command will be displayed in a new tab.	All hierarchies
	Click this button to export a hierarchy into a text file in the specified location.	All hierarchies
	Click this button to close the selected tab of results.	All hierarchies
	Click this button to show reference page.	All hierarchies

Inspection Tool Window

View | Tool Windows | Inspection

The Inspection tool window displays inspection results on separate tabs.

The left-hand pane of each tab shows a tree view of the inspections for which problems are found. The right-hand pane shows summary information for an item selected in the left-hand pane.

- You can access the tool window this way only when it is already opened through Code | Inspect Code.
- After you deactivate the tool window manually by clicking the Close button, the tool window is again available only through Code | Inspect Code/

On this page:

- Toolbar buttons
- Context menu commands
- Inspection report

Toolbar buttons

Item	Shortcut	Description
	Ctrl+F5	Click this button to run the inspection and show the results on the same tab.
	Ctrl+Shift+F4	Click this button to close the current tab or the tool window.
	Ctrl+NumPad Plus	Expand all nodes
	Ctrl+NumPad -	Collapse all nodes
	Ctrl+Alt+Down	Navigate to the next item.
	Ctrl+Alt+Up	Navigate to the previous item.
	F1	Use this icon or shortcut to open the corresponding help page.
		If this toggle is on, the problems are grouped into Errors and Warnings. Otherwise, the problems are grouped by inspections.
		Turns grouping by directories on or off.
		When this toggle is on, the resolved and excluded items are not shown.
		Turns the Autoscroll to source option on or off. When the option is on and you select an item, the corresponding source file opens in the editor and the appropriate fragment is highlighted.
		Click this button to export the inspection results into XML or HTML format .
		Click this button to edit the current inspection settings .
	Alt+Enter	Click this button to resolve the problem for the selected inspection item by choosing one of the available quick fixes from the list.

Context menu commands

Item	Shortcut	Description
Jump to Source	F4	Open the file that contains the selected problem in the editor and place the cursor at the beginning of the corresponding code fragment.
Exclude	Delete	Exclude the selected items from further examination. Excluded nodes are shown strikethrough. If the filter toggle is on, the excluded nodes are hidden.
Include	Insert	Include previously excluded items in the list of results. All nested subelements are included too.
	Alt+Enter	Select one of the suggested solutions.
Suppress problem / Suppress problem for class		Suppress the inspection for the selected problem or the selected class.
Edit Settings		Change the settings for the selected inspection or group of inspections in the Errors dialog.
Disable inspection		Disable alerts for the selected inspection in the active tab of results. If the filter toggle is on, the nodes for disabled inspections are hidden. See also, Disabling and Enabling Inspections .
Run inspection on		Rerun the selected inspection and display the results on a new tab.

Inspection report

The inspection report is shown in the right-hand pane of the results tab when an inspection node is selected in the left-hand pane. The report may include the following:

- Problem resolution: A button for each of the available solutions. Clicking a button invokes the corresponding fix. If no buttons are present, you have to fix the problem yourself.
- Suppress: Click this button to reveal the list of inspection suppress options.
- Problem synopsis: A brief description of the problem.
- Disable inspection: Disable alerts for the selected inspection in the active tab of results. If the filter toggle is on, the nodes for disabled inspections are hidden.
  See also, Disabling and Enabling Inspections .
- Run inspection on...: Rerun the selected inspection and display the results on a new tab.

JSTestDriver Server Tool Window

View | Tool Windows | JSTestDriver Server

- Start a local server link.

In this tool window, start and stop the JSTestDriver Server to run unit tests against and capture the browser to execute unit tests in.

Prerequisites

The tool window becomes available when the following prerequisites are met:

1. The JSTestDriver plugin is downloaded, installed, and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
2. The current project contains at least one test runner configuration files .

Options

Item	Tooltip and shortcut	Description
	Run a local server	Click this button to have PhpStorm launch the default JSTestDriver server.
	Stop the local server	Click this button to have PhpStorm stop the currently running JSTestDriver server.
Capture a browser using the URL		This read-only field shows the URL address to access the Remote Console of the JSTestDriver. Copy the URL address and open it in the browser of your choice.
		The icons indicate available browsers. The icon that corresponds to the browser you just opened, is active. Click the icon to get ready for executing tests.

NPM Tool Window

Context menu of a package.json - Show npm Scripts

View | Tool Windows | npm

- the tool window can be accessed this way only after you have opened it using the Show npm Scripts command.

On this page:

- Accessing the npm Tool Window
- Building a Tree of npm Scripts
- Running npm Scripts
- Toolbar
- Context Menu of a Tree
- Context Menu of a Script

Accessing the npm Tool Window

The tool window is available only when:

1. The Node.js runtime environment is installed on your computer.
2. The NodeJS repository plugin is installed and enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .
3. At least one `package.json` file is available in the current project.

The tool window opens when you invoke npm by choosing Show npm Scripts on the context menu of a `package.json` in the Project tool window or of a `package.json` opened in the editor. Use the tool window to run npm scripts.

As soon as you invoke npm, it starts building a tree of scripts defined within the `scripts` property of the `package.json` file on which it was invoked. If you have several `package.json` files in your project, you can build a separate script tree for each of them and run scripts without abandoning the previously built trees. Each tree is shown under a separate node.

Building a Tree of npm Scripts

To build a tree of scripts, do one of the following:

- Select the required `package.json` file in the Project tool window and choose Show npm Scripts on the context menu of the selection.
- Open the required `package.json` file in the editor and choose Show npm Scripts on the context menu of the editor.
- If the npm tool window is already opened, click on the toolbar and choose the required `package.json` file from the list. PhpStorm adds a new node and builds a scripts tree under it. The title of the node shows the path to the `package.json` file according to which the tree is built.

Running npm Scripts

To run a script from the tree, do one of the following:

- Double click the required script in the tree.
- Select the required script and choose Run <script name> on the context menu of the selection.
- Select the required script and press `Enter`.

The tool window shows the npm script output, reports the errors occurred, lists the packages or plugins that have not been found, etc. The name of the last executed script is displayed on the title bar of the tool window.

To run several scripts, use the multiselect mode: hold `Shift` (for adjacent items) or `Ctrl` (for non-adjacent items) keys and select the required scripts, then choose Run on the context menu of the selection.

Toolbar

Item	Tooltip	Description
	Add package.json	Click this button to have a tree of scripts for another `package.json` file built. Choose the required `package.json` file from the pop-up list. PhpStorm adds a new node and builds a tree of scripts under it it.
	Remove package.json	Click this button to remove the tree of scripts under the selected node.
	Reload scripts	Click this button to have the tree of scripts under the selected node re-built. You may need a tree re-built after updating the corresponding `package.json` file because npm does not apply changes to trees on the fly.

Context Menu of a Tree

Item	Description
npm Settings	Choose this menu item to open the npm Settings dialog box and view or edit the Node.js configuration
Jump to Source	Choose this menu item to open the `package.json` file for which the current tree is built.
Reload scripts	Choose this menu item to have the tree of scripts under the selected node re-built.
Copy Path	Choose this menu item to save the path to the `package.json` file according to which the current tree was built to the clipboard.
Remove package.json	Choose this menu item to remove the tree of scripts under the selected node.

Context Menu of a Script

Item	Description
Run <script name>	Choose this menu item to run the selected script.

Edit <script name> settings	Choose this menu item to open the Run/Debug Configuration dialog box and edit the predefined settings for the selected script.
Jump to Source	Choose this menu item to open the `package.json` file for which the current tree is built and navigate to the definition of the selected script.

Messages Tool Window

View | Tool Windows | Messages

`Alt+0`

PhpStorm parses the output and displays it in a convenient format in the Messages window, letting you navigate to the source of the problem whenever possible.

> **Note** The Messages tool window is only available if there are actually some messages to show.

On this page:

– Toolbar Buttons

– Results Context Menu

Toolbar Buttons

Item	Tooltip and shortcut	Description
	Rerun `Ctrl+F5`	Click this button to rerun compilation. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.
	Rerun Pub Command	Click this button to run the last executed Pub command. The button is available only in Dart projects, see Managing Dart Packages and Assets for details.
	Pause output	Click this button to pause the compilation process. This button is enabled when compilation is in progress. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.
	Stop/Stop Pub Process	Click this button to terminate compilation or Pub process. This button is enabled when compilation or a process is in progress.
	Close `Ctrl+Shift+F4`	Click this button to terminate the process and close the console.

	Previous/Next Message
	`Ctrl+Alt+Up`
	`Ctrl+Alt+Down`
		Navigate to the previous/next message.
		The buttons are not available in Dart projects, see Managing Dart Packages and Assets for details.

	Export to Text File `Alt+O`	Save the current console contents. In the Export dialog, specify the target file or copy information to the Clipboard. Before saving, you can also edit the information to be saved. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.
	Toggle tree/text mode	If this button is pressed, the output is displayed as plain text. If this button is released, the output is displayed as a tree view. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.
	Show all messages	When this button is pressed, the only error messages are displayed. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.

	Expand all
	`Ctrl+NumPad Plus`
	Collapse all
	`Ctrl+NumPad -`
		Use these buttons to have all nodes expanded or collapsed.
		The buttons are not available in Dart projects, see Managing Dart Packages and Assets for details.

	Hide warnings	When this button is pressed, the tree of messages shows error messages only. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.
	Autoscroll to source	If this button is pressed, the file that contains the selected error automatically opens in the editor, with the caret at the appropriate line. The button is not available in Dart projects, see Managing Dart Packages and Assets for details.

Results Context Menu

The menu items are not available in Dart projects, see Managing Dart Packages and Assets for details.

Item	Shortcut	Description

Jump to source	F4	Navigate to the selected item in the editor.
Copy	Ctrl+C	Take the line at caret to the clipboard.

Phing Build Tool Window

On this page:

- Accessing the Phing Build Tool Window
- Toolbar
- Context Menu

Accessing the Phing Build Tool Window

- Open a Phing build file in the editor or select it in the Project tool window, and then choose Add as Phing build file on the context menu of the selection.
- Choose View | Tool Windows | Phing Build on the main menu. The tool window can be accessed after you have opened it through the context menu of a Phing build file in the editor or in the Project tool window.

In this tool window:

- Specify the location of the `phing.bat` file.
- Configure a list of Phing build files for packaging, deploying, or testing PHP applications.
- Appoint targets to be executed before running or debugging according to specific configurations.
- Run entire build files and specific build targets.

The functionality of the Phing tool window is available through the toolbar buttons> and the context menu of a build file or target.

Toolbar

Item	Tooltip and shortcut	Description
	Add	Click this button add a new script to the list. In the Select Phing Build Script dialog box, that opens, choose the desired file.
	Remove	Click this button to remove the selected build file from the list.
	Run	Click this button to run the selected build file or target.
	Expand all `Ctrl+NumPad Plus`	Click this button to have PhpStorm show a tree of targets defined in the selected build file.
	Collapse all `Ctrl+NumPad -`	Click this button to have PhpStorm collapse all the targets in the selected build file.
	Properties	Click this button to have the Phing Properties dialog box opened. In this dialog box, specify the location of the `phing.bat` file. If necessary, configure the build procedure and the format of the output by specifying command line arguments in the Command Line Options text box. If the set of arguments is too large to fit in the text box, click the button or press `Shift+Enter` and type the desired arguments in the Command Line Options dialog box, that opens. The button is only available if the list of build files is not empty.

Context Menu

Item	Description	Available for
Run Build	Choose this option to have the selected build file executed.	Build files
Run Target	Choose this option to have the selected build target executed.	Build targets
Jump to Source	Choose this option to navigate to the source code of the selected build file or to the definition of the selected target.	Build files Build targets
Remove	Choose this option to remove the selected build file from the list.	Build files
Before Run/Debug	Choose this option to have the selected build target always executed before running or debugging the project according to specific run/debug configurations. In the Execute Target Before Run/Debug dialog box that opens, select the configurations before which you want the target executed.	Build targets
Assign shortcut	Choose this option to open the Keymap dialog box and configure a keyboard shortcut for the selected target as described in Configuring Keyboard Shortcuts .	Build targets
Mark to hide	Choose this option to make the selected target hidden and thus suppress showing it in the build file tree. This may be helpful when you have targets some that are only called by other ones and are never run alone. Hidden targets do not become visible when you expand the node of a specific build file or click Expand All button . To remove the hidden status of a target: 1. Select the build file in which it is defined and click the Settings button on the toolbar. 2. In the Phing Settings dialog box that opens, switch to the Hiding targets tab where the Hide check boxes next to the names of the hidden targets are selected. Clear the Hide check boxes next to the targets for which you want the status hidden removed.	Build targets
Properties	Click this button to have the Phing Properties dialog box opened. In this dialog box: - Specify the location of the `phing.bat` file. - If necessary, configure the build procedure and the format of the output by specifying command line arguments in the Command Line Options text box. If the set of arguments is too large to fit in the text box, click the button or press	Build files build

`Shift+Enter` and type the desired arguments in the Command Line Options dialog box, that opens.

targets

- In the Hiding Targets tab, mark targets for hiding and remove the hidden status.
  Among the targets defined in a build file, you may have some that are only called by other targets and are never run alone. You can suppress showing such targets in the build file tree by marking them as hidden.

  Hidden targets do not become visible when you expand the node of a specific build file or click Expand All button .

The button is only available if the list of build files is not empty.

Phing Settings Dialog

PhpStorm editor | Context menu of a Phing build file | Add as Phing build file

Project Tool Window | Context menu of a Phing build file | Add as Phing build file

View | Tool Windows | Phing Build

The dialog box opens when you click the Settings button on the toolbar of the Phing Build tool window.

In this dialog box:

- Enable Phing integration in the current project.
- Configure Phing build properties externally to pass them through a command line.
- Show/hide targets in a specific file.

On this page:

- General Area
- Properties Tab
- Hiding Targets Tab

General Area

Item	Description
Path to Phing executable	In this text box, specify the location of the `phing.bat` file. Type the path manually or click the Browse button and choose the file location in the dialog box that opens.
Command line options	In this text box, specify the additional command line arguments to be passed to Phing when the build is launched. If necessary, click the and type the desired options in the Command Line Options dialog box. Type each option on a new line. When you close the dialog box, they are all displayed in the Command line options text box with spaces as separators.

Properties Tab

Use this tab to configure a list of property elements externally. These properties are passed to Phing through a command line when build execution is launched and therefore they are every time re-calculated dynamically.

Item	Description
Add	Click this button to have an empty line for a new property element added tot he list.
Remove	Click this button to remove the selected element from the list.
Property	In this text box, type the name of the property element.
Value	In this text box, specify the value of the element to be passed during build execution. Do one of the following: – Type the value manually. – To use PhpStorm macros , click the Insert macro button and configure a list of relevant macro definitions in the Macros dialog box that opens. To add a macro, select it in the list and click Insert.

Hiding Targets Tab

In this tab, specify which targets defined in the currently selected build file you want to have shown or hidden in the file tree view.

Among the targets defined in a build file, you may have some that are only called by other targets and are never run alone. You can suppress showing such targets in the build file tree by marking them as hidden.

Hidden targets do not become visible when you expand the node of a specific build file or click Expand All button .

Item	Description
Target	This field shows a list of all the targets defined in the current build file.
Hide	– When this check box next to the name of a target is selected, the target is not shown in the tree view of the file. – When this check box is cleared, the corresponding target is displayed in the tree view of the file.

You can mark a target as hidden both directly from the Phing Build tool window or from the Phing Settings dialog box. However, the hidden status can be removed only through the Phing Settings dialog box.

Project Tool Window

View | Tool Windows | Project

Alt+1

The Project tool window lets you look at your project from various viewpoints and perform the tasks such as creating new items (directories, packages, files, classes, etc.), opening files in the editor, navigating to the code fragment of interest, and more.

Most of the functions in this tool window are accessed as context menu commands in the content pane and associated shortcuts.

- Views
- Context menu for the title bar
- Title bar icons
- Content pane
- Context menu commands for the content pane items
- File status highlights

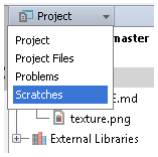
Views

The tool window provides a number of views.

Different views emphasize different project aspects and, generally, define which items are shown and how:

- Project view. In this view, all the project items along with their dependencies (SDKs and libraries) are shown. The emphasis is on the directory structure (though the packages are also shown).
- Scopes views (Project Files, Problems, etc.). What is shown in the content pane is limited to the corresponding predefined or user-defined scope . In other respects, depending on the currently selected view options, a scope view may resemble the Project view.
- Scratches view. This view lets you manage your scratch files and database consoles .

The necessary view is selected from the list in the left-hand part of the title bar or, if the views are represented by tabs, by clicking the corresponding tab.



To configure a view, use the corresponding options in the title bar context menu . The necessary options can also be accessed by clicking on the title bar.

Context menu for the title bar

The title bar context menu provides the options that let you control every aspect of the tool window appearance and behavior.

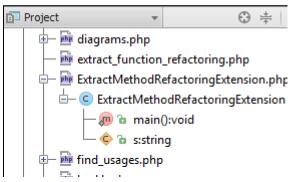
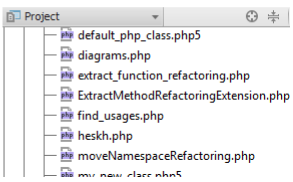
There are the options for configuring the project views . There are also the ones for controlling the tool window viewing modes .

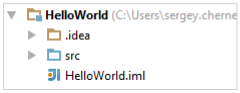
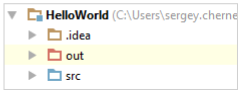
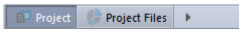
The options that are on, have a check mark to the left of their names.

The menu also includes commands for switching between the views, resizing the tool window, and more.

To access the menu, right-click the list of views in the left-hand part of the title bar, or, if the views are represented by tabs, right-click the tab of interest.

The following table lists and briefly explains the available commands and options.

Item	Shortcut	Description
Select Next View or Tab	Alt+Right	These are the commands for switching the views.
Select Previous View or Tab	Alt+Left	
Show List of Views or Tabs	Alt+Down	
Edit Scopes		Use this command to open the Scopes dialog in which you can create and edit used-defined scopes . Note that this command is available only if the current view is a scope view.
Show Members		If this option is on, the files in the tree that contain classes turn into nodes. When such node is unfolded, the contained classes with their fields, methods, and other members of the selected item are shown. **Show Members on**  **Show Members off** 
Autoscroll to Source		If this option is on, PhpStorm automatically navigates from a file (or a class member) selected in the Project tool window to the corresponding source file (or its fragment) in the editor. If the corresponding file is not currently open, it will open automatically.
Autoscroll from Source		If this option is on, PhpStorm automatically navigates from a file in the editor to the corresponding node (file, class, field, method, etc.) in the Project tool window.
Sort by Type		If the option is off, the items (files, classes, etc.) are sorted alphabetically. If the option is on, the files are sorted by their extensions.

		Folders are always sorted alphabetically, files within are shown according to the status of this option.
Folders Always on Top		If the option is on, all the folders are shown before the files. Otherwise, all the items are sorted alphabetically, and the files and folders appear intermixed.
Show Excluded Files		This option is available only in the Project view. (In other views, excluded files are never shown.) Turn this option on or off to show or hide excluded folders and files . Show Excluded Files: OFF  Show Excluded Files: ON 
Pinned, Docked, Floating, Windowed, Split Mode		These options let you control general appearance and behavior of the tool window. See Viewing Modes .
Remove from Sidebar		This command hides the tool window, removes the associated tool window button from the tool window bar and removes the tool window from the quick access menu (or). To open the tool window again (and restore the associated features), use the main menu: View | Tool Windows | <Window Name>.
Group Tabs		If this option is on, there is a list in the left-hand part of the title bar from which you can select the necessary view. If this option is off, the views are represented by tabs which appear in the left-hand part of the title bar. Group Tabs: ON Group Tabs: OFF 
Move to		To associate the tool window with a different tool window bar , select this command, and then select the destination tool window bar (Top, Left, Bottom or Right).
Resize		To resize the tool window by moving one of its borders, select this command, and then select the necessary Stretch to option. Note that this command is not available for the floating mode.
Hide	Shift+Escape	Use this command to hide the tool window.

Title bar icons

ItemShortcutDescription		
	Alt+Right Alt+Left	If the views are currently shown as tabs (the Group Tabs option is off), this button appears to the right of the last visible tab. If the first or the last of the available views is currently selected, this button is shown as or . Click this button to open the list of views, for example, to select a different view.
		Click this icon to navigate from a file in the editor to the corresponding node (file, class, field, method, etc.) in the Project tool window. This icon is not available if the Autoscroll from Source option is currently on.
	Ctrl+NumPad -	Use this icon or shortcut to collapse all the nodes.
		Click this button to open the menu for configuring the current view and changing the tool window viewing modes . Note that most of the menu items are options that you can turn on or off. An option which is on has a check mark to the left of its name. The available options are a subset of the title bar context menu items. Depending on the current view, the menu may include the following options: – Edit Scopes – Show Members – Autoscroll to Source – Autoscroll from Source – Sort by Type – Folders Always on Top – Show Excluded Files – Pinned Mode, Docked Mode, Floating Mode, Windowed Mode, Split Mode – Remove from Sidebar – Group Tabs – Move to – Resize
	Shift+Escape	Use this icon or shortcut to hide the tool window. When used in combination with the Alt key, clicking this icon hides all the tool windows attached to the same tool window bar .

Content pane

The content pane shows the project items such as modules, packages, directories, files, classes, libraries, etc.

The icons for the main categories (node types) are shown and briefly explained in the following table. The icons used for the main file types are listed in File Types Recognized by PhpStorm ; the icons for the main symbols (classes, fields, methods, etc.) are shown in Symbols .

ItemDescription	
	A package.
	A folder (directory). Different folder types have different colors: – An "ordinary" folder . – A source folder . – A test source folder . For more information on folder types, see Content Root .

External Libraries, a category for grouping third party software associated with the project. The node is only shown when the Project view is selected.

Context menu commands for the content pane items

When you right-click an item in the content pane, the context menu for this item is shown. This menu provides access to all the functions available for the selected item.

The following table lists and briefly explains the most frequently used context menu commands.

Item	Shortcut	Description
New	Alt+Insert	Use this command to create a new item (directory, file, or class). within the selected one (project or directory).
Cut	Ctrl+X	Use this command to move the selected item or items from the current location to the clipboard.
Copy	Ctrl+C	Use this command to copy the selected item or items to the clipboard.
Copy Path(s)	Ctrl+Shift+C	Use this command to copy the full path(s) of the selected item or items to the clipboard.
Copy Reference	Ctrl+Shift+Alt+C	Use this command to copy a reference to the selected item to the clipboard.
Paste	Ctrl+V	Use this command to insert the contents of the clipboard into the selected location.
Jump to Source	F4	Use this command to open the selected file in the editor. If the file is already open, the corresponding editor tab will become active.
Find Usages	Alt+F7	Use this command to find the usages of the selected item. (The Find Usages dialog will open.)
Find in Path	Ctrl+Shift+F	Use this command to perform a text search. (The Find in Path dialog will open.)
Replace in Path	Ctrl+Shift+R	Use this command to perform text search-and-replace. (The Replace in Path dialog will open.)
Refactor		Use this command to perform one of the refactorings available for the selected item.
Add to Favorites		Use this command to add the selected item to an existing or new list of favorite items. See Managing Your Project Favorites .
Show Thumbnails	Ctrl+Shift+T	Use this command to view thumbnails for image files located in the selected directory. (The Thumbnails tool window will open.)
Reformat Code	Ctrl+Alt+L	Use this command to reformat the source code in the selected file or in all files in the current directory. (The Reformat Code dialog will open.) See also, Reformatting Source Code .
Delete	Delete	Use this command to delete the selected item. Use with care!
Change Dialect (<CurrentDialect>)		For SQL files and database consoles: change the SQL dialect associated with the file or console.
Run '<item_name>'	Ctrl+Shift+F10	For an SQL file or database console: execute all the statements contained in the selected file or console.
Local History		Use this command to view local history for the selected file or directory, or to create a label for the current version of your project. See Local History and Using Local History .
Synchronize '<item_name>'		Use this command to synchronize the selected item with its version saved in the file system. (If you change a file or directory contents externally, PhpStorm, under certain circumstances, may not be aware of the corresponding changes unless you use this command.)
Show in Explorer		Use this command to open a file browser (e.g. Windows Explorer or Finder) and show the selected item there.
File Path	Ctrl+Alt+F12	Use this command to open the File Path menu. This menu shows the path from the file system root to the selected element with individual directories as the menu items. When you select an item in this menu (e.g. a directory), a file browser (e.g. Windows Explorer or Finder) opens, and the selected item is shown there.
Compare With	Ctrl+D	Use this command to compare the selected file or directory with another file or directory. Select the other file or directory the dialog that opens . See Comparing Folders and Differences viewer for folders .
Compare File with Editor		Use this command to compare the selected file with the file open on an active editor tab. See Comparing Files and Differences viewer for files .
Mark Directory As		Use this command to make the selected directory a source root or a test source root , to make the directory excluded, etc. The necessary category for the directory is selected from the submenu. For more information on the categories for directories, see Content Root .
Diagrams	Ctrl+Shift+Alt+U or Ctrl+Alt+U	Use this command to open a diagram (e.g. a UML diagram) for the selected item. For more information, see Diagram Reference .

File status highlights

PhpStorm uses colors to denote VCS file status in the Project tool window. The following table presents information about the meaning of the colors.

Color	File Status	Description
Black	Up to date	File is unchanged. test1.txt
Gray	Deleted	File is scheduled for deletion from the repository. privatent.txt
Blue	Modified	File has changed since the last synchronization. test1.txt

Green	Added	File is scheduled for addition to the repository. test1.txt
Violet	Merged	File is merged by your VCS as a result of an update. test1.txt
Brown	Unknown	File exists locally, but is not in the repository, and is not scheduled for adding. Test.xml
Olive	Ignored	File will be ignored in any VCS operation. Test.html
Light brown	Hijacked	File is modified without checkout . This status is valid for the files under Perforce, ClearCase and VSS. HelpTOC.xml
Red	Merged with conflicts	During the last update, file was merged with conflicts. test1.txt
Lilac	Externally deleted	File is deleted locally, but was not scheduled for deletion, and still exists in the CVS repository. test1.txt
Dark cyan	Switched	The file is taken from a different branch than the whole project. This status is valid for CVS and SVN. test1.txt

Remote Host Tool Window

Tools | Deployment | Browse Remote Host

View | Tool Windows | Remote Host

Use this tool window to view the folder structure of the target FTP/FTPS/SFTP servers and the data uploaded to them.

Getting Access to the Remote Host Tool Window

- View | Tool Windows | Remote Host - the tool window can be accessed this way after you have opened it using the Tools | Deployment | Browse Remote Host command.
- The tool window is available only when the Remote Hosts Access bundled plugin is enabled. The plugin is active by default. If not, activate it in the Plugins page of the Settings dialog box.

Toolbar

Item	Tooltip and Shortcut	Description
Remote host		From this drop-down list, select the desired remote host configuration.
	Shift+Enter	Click the browse button to add a new server .
	Collapse All Ctrl+NumPad -	Click this button to have all nodes in the view collapsed.
	Refresh Ctrl+F5	Click this button to refresh the view.
	Close Ctrl+Shift+F4	Click this button to close the tool window.
	Help F1	Click this button to show this reference page.

Context Menu

Item	Description
Upload Here	Choose this option to have the files from the currently opened project uploaded according to the selected configuration. If the action is invoked from the context menu of a file, the corresponding local file is uploaded. If the action is invoked from a folder, the entire folder is uploaded.
Download from here	Choose this option to download the selected file or folder to the currently opened. The existing local files will be updated and the missing files will be created.
New	Choose this option to create a new remote file or folder in the selected folder. The option is available only from the context menu of a folder.
Rename	Choose this option to rename the selected file or folder.
Delete	Choose this option to remove the selected file or folder.
Cut	Choose this option to copy the name of the selected file or folder to the clipboard and remove the file or folder from the tree.
Copy	Choose this option to copy the name of the selected file or folder to the clipboard.
Paste	Choose this option to insert the previously copied name of a file or a folder into the tree.
Edit Local File	When you select a file and choose this option, PhpStorm opens its local copy in the editor and moves the focus to the corresponding. The option is available only when the project with the corresponding local file is currently opened i the editor.
Sync with Local	Choose this option to compare the selected remote folder with its local version. In the Differences Viewer for Folders that opens, explore the differences and synchronize the files , where applicable. See Comparing Deployed Files and Folders with Their Local Versions for details.
Compare with Local Version	Choose this option to compare the selected remote file with its local version. In the Differences Viewer for Files dialog box, that opens, explore the differences and apply them, if necessary, using the » and « buttons. For details, see Viewing Differences Between Files . See Comparing Deployed Files and Folders with Their Local Versions for details.
Edit Remote File	Choose this option to edit the selected file in the PhpStorm editor without adding it to the currently opened project. See Editing Individual Files on Remote Hosts for details.
Copy Path	
Exclude Path	Choose this option to exclude the selected folder from upload/download, see Excluding Files and Folders from Upload/Download for details.
Copy Path	Choose this option to copy the absolute path to the selected file or folder on the server to the clipboard.

REST Client Tool Window

Tools | Test RESTful Web Service

View | Tool Windows | REST Client

Use the REST Client tool window for testing a RESTful Web Service . The tool window is intended for composing and submitting test requests to Web service methods based on the service API, as well as for viewing and analyzing server responses.

Please note the following:

- View | Tool Windows | REST Client - the tool window can be accessed this way after you have opened it using the Tools | Test RESTful Web Service command.
- The tool window is available only when the REST Client bundled plugin is enabled. The plugin is active by default. If not, activate it in the Plugins page of the Settings dialog box.

On this page:

- Common Request Settings
  - Toolbar
- Request Tab
- Cookies Tab
- Response Tab
- Response Headers Tab

Common Request Settings

In this area, choose the request method and specify the data to compose the request URI from.

The server response code and the content length are shown in the upper-right corner of the REST Client tool window.

Item	Description
HTTP method	In this drop-down list, specify the request method . The available options are: - GET - POST - PUT - PATCH - DELETE - HEAD - OPTIONS
Host/port	In this text box, type the URL address of the host where the target Web service is deployed and the port it listens to. By default, the port number is `80` . If another port is used, specify it explicitly in the format `<host URL>:<port number>` .
Path	In this drop-down list, specify the relative path to the target method.

You can enter the entire URL address of a method to test in the Host/port text box. Regardless of the chosen HTTP method, upon pressing PhpStorm will split the URL address into the host/port and the path to the method. The extracted relative path will be shown in the Path text box and the extracted parameters will be added to the list in the Request Parameters pane of the Request tab.

Toolbar

Item	Tooltip and shortcut	Description
	Submit Request	Click this button to submit the generated test request to the server. Note If a server is not trusted, PhpStorm shows a dialog box suggesting you to accept the server, or reject it. If you accept the server as trusted, PhpStorm writes its certificate to the trust store. On the next connect to the server, this dialog box will not be shown.
	Replay Recent Requests	Click this button to have a Recent Requests pop-up list displayed and select the relevant request. The fields are filled in with the settings of the selected request. Click the Submit Request button .
	Export Request	Click this button to have the current request settings saved in an XML file so they are available in another PhpStorm session. In the dialog box that opens, specify the name of the file to save the settings in and its parent folder. When necessary, you can retrieve the saved settings and run the request again.
	Import Request	Click this button to have the settings of a previously saved request retrieved from an XML file. In the dialog box that opens, select the relevant XML file.
	Generate Authorization Header	Click this button to open the Generate Authorization Header dialog box and specify your user name and password for accessing the target RESTful Web service through. Based on these credentials PhpStorm will generate an authentication header which will be used in basic authentication . Learn more at http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm .
	Configure HTTP Proxy	Click this button to specify proxy server settings in the dialog box that opens.
	Close	Close the REST Client tool window.
	Help	Show this page.

Request Tab

Use this tab to specify the parameters to be passed to the service in the generated test request either through the query string for `GET` requests or through the request body for other request types. Also configure interaction between the client side and the Web service by specifying the format of data that the service and the client accept.

Item	Description
Headers	In this pane, specify the technical data included in the request header . These data are passed through header fields and define the format of the input parameters (accept field), the response format (content-type field), the caching mechanism (cache-control field), etc. To add a field to the list, click Add +, then specify the field name in the Name text box and the field value in the Value drop-down list. The set of fields and their values should comply with the Web service API. In other words, the specified input format should be exactly the one expected by the Web service as well as the expected response format should be exactly the one that the service returns. For `accept` , `content-type` , and some other fields PhpStorm provides a list of suggested values. Choose the relevant format type from the Value drop-down list.
Request Parameters	In this pane, specify the parameters to be passed to the target method through a query string inside the URL. This approach is used for requests of the type `GET` . By default, the pane shows an empty list with one line. – To add a parameter, click Add +, then specify the name of the parameter in the Name text box and the value of the parameter in the Value drop-down list. – To delete a parameter from the list, select it and click Remove –. – To suppress sending the specified query string parameters and disable the controls in the Request Parameters pane, press the Don't send anything toggle button . The set of parameters and their types should comply with the Web service API, in particular, they should be exactly the same as the input parameters of the target method. – select this check box This may be helpful, for example, if you want to test passing parameters through other request methods in the request body but still preserve the data typed in the Query string parameters pane.
Request Body	The pane is disabled when the `GET` , `DELETE` , `HEAD` , or `OPTIONS` request method is selected. In this pane, specify the input parameters to be passed to the target method inside a request message body . – Empty: choose this option to send a request with an empty body. – Text: choose this option to send a request with a string of parameters with values. Specify the parameters in the text box next to the option. – File contents: choose this option to have the parameters inserted in the request body from a local text file. Specify the source text file in the File to send field. – File upload(multipart/form-data): choose this option if you want to pass a binary file as a parameter, which requires that the file be converted. Specify the source binary file in the File to send field.

Cookies Tab

Use this tab to create, save, edit, and remove cookies, both received through responses and created manually. The name and value of a cookie is automatically included in each request to the URL address that matches the domain and path specified for the cookie, provided that the expiry date has not been reached.

The tab shows a list of all currently available cookies that you received through responses or created manually. The cookies are shown in the order they were added to the list. When you click a cookie, its details become editable and are displayed in text boxes.

Item	Description
Name	In this fields, specify the name of the cookie to be included in the request.
Value	In this field, specify the value of the cookie to be included in requests.
Domain	In this field, specify the host and port the requests to which must be supplied with the name and value of the cookie.
Path	In this field, specify the path of the URL the requests to which must be supplied with the name and value of the cookie.
Expiry date	In this field, specify the expiry date of the cookie.
+	Click this button to add a new row to the list and define a new coolie in it.
–	Click this button to remove the selected cookie from the list.

Response Tab

Use this tab to view responses from the Web service. By default, responses are shown in the plain text form. Use the icons of the tab to have them displayed in the editor in the HTML, XML, and JSON formats.

Item	Tooltip	Description
	View as HTML	Click this button to open a new tab in the main editor window and display there the response as HTML.
	View as XML	Click this button to open a new tab in the main editor window and display there the server response as XML.
	View as JSON	Click this button to open a new tab in the main editor window and display there the server response as JSON.
	Open in browser	Click this button to view the response in your default Web browser.

Response Headers Tab

The tab shows the technical data provided in the headers of Web service responses .

Run Tool Window

Run | Run

View | Tool Windows | Run

`Alt+4`

The Run tool window displays output generated by your application. If you are running multiple applications, each one is displayed in a tab named after the run/debug configuration applied.

The appearance of each tab depends on the type of the application being run and can include additional toolboxes and panes.

- Run Tool Window
  - Run Toolbar
  - Context Menu Options
  - Console Toolbar
  - Karma Server tab
- Test Runner Tab
- Export Test Results

The main toolbar of the Run tool window lets you rerun, stop, pause, or terminate an application. The following table contains descriptions of the buttons that are common for most applications.

Run Toolbar

Item	Tooltip and shortcut	Description
	Rerun `Ctrl+F5`	Click this button to stop the current application and run it again. When an application is stopped ( ), this button toggles to .
	Rerun `Ctrl+F5`	Click this button to rerun the current application. This button appears, when an application is stopped ( ). When an application is running, this button toggles to .
	Pause Output	Click this button to have the process output paused. Note that the button is not available for Run/Debug Configuration: Node JS , Run/Debug Configuration: Node JS Remote Debug , and Run/Debug Configuration: NodeUnit .
	Stop `Ctrl+F2`	Click this button to terminate the current process externally by means of the standard `shutdown` script. For a Node.js run configuration , clicking the button once invokes soft kill allowing the Node.js application to catch the `SIGINT` event and perform graceful termination (on Windows, the `Ctrl+C` event is emulated). After the button is clicked once, it is replaced with indicating that subsequent click will lead to force termination of the application, e.g. on Unix `SIGKILL` is sent.
	Restore Layout	Click this button to to have the changes to the current layout abandoned and return to the default state.
	Pin	Use to pin or unpin the tab. If a tab is pinned, the results for the next command are shown on a new tab.
	Close `Ctrl+Shift+F4`	Click this button to close the selected tab of the Run tool window and terminate the current process.
	Help `F1`	Use this icon or shortcut to open the corresponding help page.

Context Menu Options

Item	Description
Compare with Clipboard	Opens the Clipboard vs Editor dialog box that allows you to view the differences between the selection from the editor and the current clipboard content. This dialog is a regular comparing tool that enables you to copy the line at caret to the clipboard, find text, navigate between differences and manage white spaces.
Copy URL	Choose this command to copy the current URL to the system clipboard. This command only shows on a URL, if it is included in an application's output.
Create Gist	Choose this command to open the Create Gist dialog box.
Clear All	Clears the output window.

Console Toolbar

Item	Tooltip and shortcut	Description

	Up/down the Stack Trace
	`Ctrl+Alt+Up`
	`Ctrl+Alt+Down`
		Click this button to navigate up or down in the stack trace and have the cursor jump to the corresponding location in the source code.

	Use Soft Wraps	Click this button to toggle the soft wrap mode of the output.
	Scroll to the end	Click this button to navigate to the bottom of the stack trace and have the cursor jump to the corresponding location in the source code.

	Print	Click this button to send the console text to the default printer.
	Clear All	Click this button to remove all text from the console. This function is also available on the context menu of the console.
	Use Console Input	– When this toggle-button is pressed, you can evaluate and alter variables, call PHP functions, and define additional functions on the fly in the Console pane. – When the toggle-button is released, any input in the Console pane will be treated like `STDIN` .

Karma Server tab

The tab is shown only when you run JavaScript unit tests using the Karma test runner. Use the tab to view and analyze information from the server. For details, see Running Unit Tests on Karma .

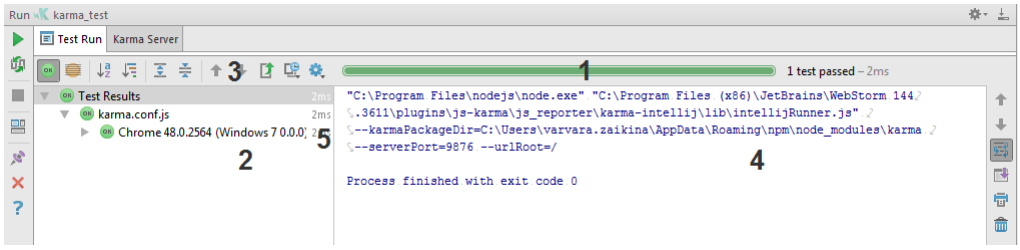
Test Runner Tab

In this section:

- Basics
- Run toolbar
- Testing toolbar
- Test status icons
- Output pane
- Context menu commands

Basics

The Test Runner tab opens in the Run tool window when a testing session begins, and features the same toolbar buttons .



1. The color of the status bar indicates whether the tests have passed successfully. If at least one of the tests fails, the status bar turns red.
2. The left-hand pane shows the tree view of all tests within the current run/debug configuration.
   - The root node represents the test case selected to run.
   - The nested nodes represent the hierarchy of test suites and test cases.
   - The leaf nodes represent the individual tests.

   The status of each test is indicated by an icon . Double-click a node to open the respective test class or test method in the editor.
3. The testing toolbar provides controls that enable you to monitor the tests and analyze results. Some of the commands are duplicated on the context menus of the test tree nodes.
4. The Output pane shows the output of the current test suit.
5. The inline statistics show the list of executed tests with the execution time of each test.

Run toolbar

ItemTooltip and Shortcut	Description	
	Rerun Ctrl+F5	Click this button to rerun the current process. The process reruns always in the same console regardless of whether this console is pinned or not.
	Rerun Failed Tests	Click this button to have PhpStorm execute failed tests. If you press Shift and click this button, you can choose whether you want to Run the tests again, or Debug, i.e. rerun the failed tests in the Debug mode.
	Stop and rerun ?	Click this button to stop executing the current test session and have the test re-started automatically. When the test execution stops, the button toggles to Rerun
	Stop Ctrl+F2	Click this button to terminate the current process externally by means of the standard mechanisms.
	Layout	Click this button to choose a layout of the toolbars by selecting the desired option from the pop-up menu.
	Pin	When this button is pressed, the current tab will not be overwritten; instead, the results of the next command will be displayed in a new tab.
	Ctrl+Shift+F4	Click this button to close the selected tab of the Run tool window and terminate the current process.
	F1	Click this button to show reference.

Testing toolbar

ItemTooltip and Shortcut	Description	
	Show Passed	Click this button to show tests that passed successfully.
	Show Ignored	Click this button to show the skipped tests in the tree view of all tests within the current run/debug configuration or test class. A test can be skipped in two situations: – Intentionally, which is defined through the @requires annotation. – When another test, on which the test in question depends, fails. Dependencies between tests are indicated through the @depends annotation.

	Sort alphabetically	Click this button to sort tests in alphabetical order.
	Sort by duration	Click this button to sort tests by duration.
	Expand All/Collapse All Ctrl+NumPad Plus Ctrl+NumPad -	Click these buttons to have all nodes in the tree view of tests expanded. These buttons are only available if the tested application contains more than test case.
	Previous/Next Failed Test Ctrl+Alt+Up / Ctrl+Alt+Down	Click these buttons to navigate between the failed tests.
	Export Test Results	Click this button to have the results of the selected test saved in a file. In the Export Test Results that opens, specify the file to save the output in and the format in which the data will be saved. If you want to view the test results later, choose the XML format. To view the saved test results later, click and specify the XML file where they are stored.
	Click this cog button to access the context menu with the following options: – Track Running Test: turn this option on to monitor execution of the current test. If a test suite contains multiple tests, the tree view of tests expands to show sequential test methods, as they are executed. – Show Inline Statistics: turn this option on to have the statistics shown next to a test result, displaying the time used for executing each test. – Scroll to Stacktrace: turn this option on to have the console scroll to the beginning of the trace of the last failed test. If you click the root node (the test package) in the tree view with this option turned off, the console will show the very beginning of the test. This option is helpful when a test package contains multiple test classes and test methods. If some of the tests fail, you can scroll in the console to the beginning of a stack trace of an exception or assertion. – Open Source at Exception: use this option to explore the results of a test that fails as an error, throwing an uncaught exception. If you double-click the failed test class or method in the tree view with this option turned on, the respective test class or method will open in the editor, with the caret placed at the line that caused the problem. – Autoscroll to Source: turn this option on to have the currently selected test in the tree view synchronized with the editor automatically. – Set Auto Test Delay: point to this node to reveal the list of available delays between file saving and rerunning tests. The selected value gets the check mark. – Select First Failed Test When Finished: turn this option on to have the first failed test automatically selected in the tree view upon completing the tests.	

Test status icons

Icon	Description
	Test error. This status is assigned to tests that caused an exception from the tested source code.
	Test failed. If at least one test receives this status, the entire tree is marked as terminated as well.
	Test ignored. In the PHP context, this icon indicates a skipped test. A test can be skipped in two situations: – Intentionally, which is defined through the @requires annotation. – When another test, on which the test in question depends, fails. Dependencies between tests are indicated through the @depends annotation.
	Test in progress.
	Test passed successfully.
	Test paused.
	Test terminated. This status is assigned to tests that were cancelled by clicking the Stop button . If at least one test receives this status, the entire tree is marked as terminated as well.
	Test not run. This status is assigned to a test when PhpStorm has not launched it at all, for example, because testing was terminated by clicking the Stop button .

Output pane

This pane shows output of each test, generated at runtime, including all the messages sent to the output stream, and the error messages. The following table shows the toolbar buttons and context menu commands available for the Output pane.

Item	Keyboard Shortcut	Description
	Up the Stack Trace Ctrl+Alt+Down	Click this button to navigate up in the stack trace and have the cursor jump to the corresponding location in the source code.
	Down the Stack Trace Ctrl+Alt+Up	Click this button to navigate down in the stack trace and have the cursor jump to the corresponding location in the source code.
	Use Soft Wraps	Click this button to toggle the soft wrap mode of the output.
	Scroll to the end	Click this button to navigate to the bottom of the stack trace and have the cursor jump to the corresponding location in the source code.
	Print	Click this button to configure printing out the console output in the Print dialog box that opens.
Clear All		Choose this item on the context menu to have all messages for the selected test deleted.

Copy Content	`Ctrl+C`	Choose this item on the context menu to have the current contents of the Output pane placed to the Clipboard.
Compare with Clipboard		Choose this item on the context menu to invoke the Differences viewer for files which shows the current contents of the Clipboard in the left-hand pane and the contents of the Output pane for the selected test in the right-hand pane.

Context menu commands

Command	Keyboard shortcut	Description
Run <test name>	`Ctrl+Shift+F10`	Run the selected test with the current temporary run/debug configuration or choose another configuration from the subordinate context menu.
Debug <test name>		Debug the selected test with the current temporary run/debug configuration or choose another configuration from the subordinate context menu.
Run <test name> with Coverage	`Ctrl+Shift+F10`	Run the selected test and collect coverage data in accordance with the current temporary run/debug configuration .
Create <test name>		Create a run/debug configuration on the base of the selected test.
Save <test name>		Save the temporary run/debug configurationn
Jump to Source	`F4`	Choose this command to move the focus to the editor, to the definition of a test class, or test method.
Show Source	`Ctrl+Enter`	Choose this command to open source code in the editor, but leave the focus with the Test Runner tab.

Export Test Results

Run tool window | Test Runner

The dialog box opens when you click the Export Test Results button in the Test Runner tab of the Run tool window .

In this dialog box, choose the format in which you want the test output saved and the file to save the test results in.

Item	Description
Export format	In this area, choose the desired output format. The available options are: – HTML – XML – Custom, apply XSL template: choose this option to have the results presented according to your own code style. Select the relevant `*.xsl` code style definition file in the file chooser.
Output	In this section, specify the target file name (File name), and directory (Folder).
Open exported file in browser	Select this check box to automatically open the above defined file with the test results in the default browser.

Spy-js Tool Window

The tool window is available only when the Spy-js plugin is enabled. The plugin is not bundled with PhpStorm, but it can be installed from the JetBrains plugin repository as described in Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins .

The tool window opens when you launch a run configuration of the type Spy-js or Spy-js for NodeJS . As soon as the Spy-js tool captures an event, it shows the event itself, its details, and stack-traces. By clicking an event, you can open the trace file, which is a prettified source file with the script that the event launched.

The tool window consists of a common toolbar and two tabs with panes:

- Trace Run Tab
- Trace Proxy Server Tab

Toolbar

Item	Tooltip and shortcut	Description
	Rerun Ctrl+F5	Click this button to stop the current trace session and run it again. When the session is canceled externally by clicking , the button toggles to . Note, that after the session restarts, you have to refresh the traced page in the browser to start capturing events.
	Rerun F9	This button substitutes for if the previous trace session was canceled externally by clicking . Click to initiate a new session. Note, that after the session restarts, you have to refresh the traced page in the browser to start capturing events.
	Stop Ctrl+F2	Click this button to terminate the current process externally by means of the standard `shutdown` script. For a Node.js run configuration , clicking the button once invokes soft kill allowing the Node.js application to catch the `SIGINT` event and perform graceful termination (on Windows, the Ctrl+C event is emulated). After the button is clicked once, it is replaced with indicating that subsequent click will lead to force termination of the application, e.g. on Unix `SIGKILL` is sent.
	Restore Layout	Click this button to to have the changes to the current layout abandoned and return to the default state.
	Pin	Use to pin or unpin the tab. If a tab is pinned, the results for the next command are shown on a new tab.
	Close Ctrl+Shift+F4	Click this button to close the selected tab of the Run tool window and terminate the current process.
	Help F1	Use this icon or shortcut to open the corresponding help page.

Trace Run Tab

The tab consists of a toolbar and three panes: Events Pane, Event Stack Pane, and Quick Evaluation Pane.

On this page:

- Events Toolbar
- Events Pane
  - Context Menu of a Document Node
  - Context Menu of an Event or Script
  - Configuring the Range of Events to Capture
    - Defining a new event filter
    - Activating a filter
    - Adding an event to an exclusion filter on the fly
- Event Stack Pane
  - Context Menu of a Script or Function
- Synchronization between the Panes and the Editor
- Quick Evaluation Pane
  - Context Menu of Function Call Details

Events Toolbar

Use the buttons on the toolbar to control the range of events to capture, configure their presentation, and navigate through the list of captured events.

Item	Tooltip and shortcut	Description
	Expand all `Ctrl+NumPad Plus`	Click this button to have all the nodes in the list expanded.
	Collapse all `Ctrl+NumPad -`	Click this button to have all the nodes in the list collapsed.
	Up the Stack Trace	Click this button to navigate to the previous traced page in the stack trace.
	Down the Stack Trace	Click this button to navigate to the next traced page in the stack trace.
	Autoscroll to source	Press this toggle button to have the list in the Events pane automatically synchronized with the Editor. – When the button is pressed: as soon as you click an event in the Events pane, the details of the event are displayed in the Event Stack pane and the script that is invoked by the event is opened in the editor automatically. When you navigate through the Event Stack with the Autoscroll to Trace mode turned on, the corresponding files are also automatically opened in the editor with the calling functions highlighted. – When the button is released: the script that is invoked by the event is opened in the editor only upon double-clicking the event in the Event Stack pane.
	Capture Events	Click this button to configure the range of events to be captured and shown in the Events list. By default, the Spy-js tool captures all events on all opened Web pages, excluding https secure web sites, unless you have specified a URL address explicitly in the run configuration. The Events pane of the Spy-js tool window shows all captured events. If for some reasons you do not want to have all events captured, you can suppress capturing some of them by applying user-defined event filters. When you click , the pop-up list shows all the available filters, the currently applied filter is marked with a tick. By default the Capture All predefined filter is applied. To stop capturing events without stopping the application, choose Mute All. The application is still running but the Events pane shows the last captured event. This is helpful if you want to analyze a script and therefore need it to be displayed in the Events pane instead of being removed as new events are captured. To define a custom event filter: 1. Click , and then choose Edit Capture Exclusions from the list. 2. In the Spy-Js Capture Exclusions Dialog that opens, click the Add toolbar button in the left-hand pane. 3. In the right-hand pane, specify the filter name in the Exclusion name field and configure a list of exclusion rules. To add a rule, click , the Add Condition to Exclusion dialog box opens. Type a pattern in the Value/pattern text box, in the Condition type drop-down list specify whether the pattern should be applied to event types or script names. Note, that glob pattern matching is used. When you click OK, PhpStorm brings you to the Spy-Js Capture Exclusions Dialog . To edit a rule, select it in the list, click , and update the rule in the dialog box that opens. To remove a rule, select it in the list and click . To activate a filter, set a tick next to the required filter in the list.
		Click this button to remove all or some events from tracing and have the corresponding trace files closed in the editor. Choose one of the following options on the drop-down list that is displayed: – Remove all: choose this option to cancel tracing of all captured events without closing the trace files in the editor. – Close all trace files: choose this option to have all trace files in the editor closed but keep tracing the corresponding event. To remove an event or script from tracing and close the corresponding trace files in the editor, choose Remove on the context menu of the event or script. – Remove all inactive: choose this option to remove all nodes for pages that are not active anymore (for example, because the pages have been closed in browser). – Save trace: choose this option to save an image of the current session. The `.json` files that store the calls and properties of the session are compressed into a `zip` archive. Upon request, when you choose Load trace, the `.json` files are extracted from the archive and loaded into Spy-js. – Load trace: choose this option to have a previously saved image of a tracing session loaded into Spy-js. The `.json` files that store the calls and properties of that session are extracted and imported. Note that a loaded image does not restore the session because no scripts are actually executed. All you can do is analyze the flow and properties of previously executed code. – Close all trace files on session stop: choose this option to have all trace files in the editor closed when you click the Stop button to stop the tracing session externally by means of the standard `shutdown` script.

For a Node.js run configuration , clicking the button once invokes soft kill allowing the Node.js application to catch the `SIGINT` event and perform graceful termination (on Windows, the `Ctrl+C` event is emulated). After the button is clicked once, it is replaced with indicating that subsequent click will lead to force termination of the application, e.g. on Unix `SIGKILL` is sent.

- Enable Spy-js autocomplete and magnifier: choose this option to have the basic completion list expanded with runtime data (Spy-js autocompletion) and to get the possibility to evaluate expressions without actually running a debugging session (Spy-js magnification). By default, the functionality is turned off.
  The term Spy-js autocompletion denotes expanding the basic completion list with suggestions retrieved from the runtime data. The Spy-js autocompletion functionality is available from source files for the code that has already been executed (highlighted green in the corresponding trace file).

  When you position the caret at a symbol in the source file and press `Ctrl+Space`, Spy-js retrieves data from the browser or from the running NodeJS application and merges it with the basic completion list according to the following rules:
  1. If an object both is present on the basic completion list and is retrieved from the runtime, the variant that provides more information about parameters, attributes, their type, etc. remains on the list.
  2. Objects retrieved by Spy.js are shown on top of the list and marked with the icon. If a retrieved object is specific for a browser, the object is marked with the icon and with the icon of this browser.

  The term Spy-js magnification denotes evaluating expressions without actually running a debugging session. When you click the expression in question or position the caret at it and press `Ctrl+Alt+F8`, a tooltip is displayed below the expression showing the expression value. If Spy-js retrieves several values, click icon in the tooltip to expand the list of values.

  The magnification functionality is available from source files for both executed and not yet executed code.
- Spy-js supports source maps, which means that you can now jump from the Event Stack pane right to the original source code in ECMAScript 6, TypeScript or CoffeeScript and observe what code fragments were executed. Use the following options to configure the way source maps are treated:
  - Enable source map look-up: choose this option to enable navigation to the ECMAScript 6, TypeScript or CoffeeScript source code using the source maps generated during transpilation.
  - Enable source map generation: choose this option to generate source maps for everything to map the instrumented code. Choose this option if you are going to debug the original code in Chrome Dev Tools or FireFox FireBug development tools.
  - Always open source mapped trace if available: choose this option to have Spy-js try to open the mapped trace file when you invoke navigation from an event to its caller.

Events Pane

The pane shows a tree of captured events. The top-level nodes represent documents that are Web pages involved in tracing. When you hover the mouse over a document, PhpStorm displays a tooltip with the URL address of the document, the browser in which it is opened, and the operating system the browser is running on. The document node is also supplied with an icon that indicates the browser in which it is opened.

Under each document node, events detected on the page and scripts started from it are listed. Events of the same type are grouped into visual containers. The header of a container displays the name of the events grouped in it, the average execution time across all the events within the container, and the number of events inside the container. You can expand each node and inspect the individual events in it.

Script file names have different colour indicators to help distinguishing between them when working with the Event Stack pane. By hovering your mouse over a script file name, you can see the full script URL.

Once an event is clicked, its call stack is displayed in the Event Stack pane. The stack is represented by a tree of function calls.

Context Menu of a Document Node

Menu item	Description
Remove	Choose this option to cancel tracing all the scripts on the selected page and remove the selected node with all the items under it from the Events pane. All the currently opened trace files remain opened in the editor.
Remove all children	Choose this option to delete the items under the selected page but keep tracing it so that new events from the page are still received. The document node itself remains in the Event pane, and all the currently opened trace files remain opened in the editor.
Remove and close trace file(s)	Choose this option to cancel tracing all the scripts on the selected page, remove the selected node and all the items under it from the Events pane, and close the corresponding trace files in the editor.
Close trace file(s)	Choose this option to close all the currently opened trace files that correspond to the selected document node and items under it. The document node and the items under it remain in the Events pane.
Refresh the page in browser	Choose this option to reload the page that corresponds to the selected document node. Tracing of the selected node is abandoned, a new document node for tracing the same page is created, and the old node becomes inactive.
Try closing the page in browser	Choose this option to close the page that corresponds to the selected node. Tracing of the selected node is abandoned, and the node becomes inactive.
Show application dependency diagram	Choose this option to build a diagram with the dependencies within the entire application. – The diagram is opened in a separate editor tab. The nodes in the diagram represent your project files, while the edges represent the fact that there's one or more functions in the source file that invoke functions in the target file. – To examine the details of a node or an edge, select the node or the edge in question and view its Details tree in a dedicated pane in the upper right-hand corner of the editor. The pane displays the connecting function combinations, along with event(s) the calls are made within and the number of calls made.

Context Menu of an Event or Script

Menu item	Description
Mute event	Choose this option to add an event to an exclusion filter on the fly.
Mute script	Choose this option to add a script to an exclusion filter on the fly.
Remove	Choose this option to cancel tracing the selected event or script, remove the selected item from the Events pane, but leave the corresponding trace files opened in the editor.
Add label	Choose this option to set a timestamp label. Timestamp labels help you to analyze your code execution within a specific period of time. For example, you can set two timestamp labels and view which events were captured between them. Or on the contrary, you can locate the events that were not captured within a certain period of time although you expected them to be and thus detect performance problems.
Show event	Choose this option to build a diagram with the dependencies in which the event selected event is involed.

dependency diagram	– The diagram is opened in a separate editor tab. The nodes in the diagram represent your project files, while the edges represent the fact that there's one or more functions in the source file that invoke functions in the target file. – To examine the details of a node or an edge, select the node or the edge in question and view its Details tree in a dedicated pane in the upper right-hand corner of the editor. The pane displays the connecting function combinations, along with event(s) the calls are made within and the number of calls made.

Configuring the Range of Events to Capture

By default, the Spy-js tool captures all events on all opened Web pages, excluding https secure web sites, unless you have specified a URL address explicitly in the run configuration. The Events pane of the Spy-js tool window shows all captured events. If for some reasons you do not want to have all events captured, you can suppress capturing some of them by applying user-defined event filters. All the available filters are listed upon clicking the Capture Events button on the toolbar, the currently applied filter is marked with a tick. By default the Capture All predefined filter is applied.

To stop capturing events without stopping the application, choose Mute All. The application is still running but the Events pane shows the last captured event. This is helpful if you want to analyze a script and therefore need it to be displayed in the Events pane instead of being removed as new events are captured.

You can define new custom filters or add event patterns to existing filters on the fly.

Defining a new event filter

1. Click the Capture Events button on the toolbar, and then choose Edit Capture Exclusions from the list.
2. In the Spy-Js Capture Exclusions Dialog that opens, click the Add toolbar button in the left-hand pane.
3. In the right-hand pane, specify the filter name in the Exclusion name field and configure a list of exclusion rules.
   To add a rule, click , the Add Condition to Exclusion dialog box opens. Type a pattern in the Value/pattern text box, in the Condition type drop-down list specify whether the pattern should be applied to event types or script names. Note, that glob pattern matching is used. When you click OK, PhpStorm brings you to the Spy-Js Capture Exclusions Dialog .

   To edit a rule, select it in the list, click , and update the rule in the dialog box that opens. To remove a rule, select it in the list and click .

Activating a filter

– Click and set a tick next to the required filter in the list. If no filters are configured or none of the available filters fits the task, create a new filter as described above.

Adding an event to an exclusion filter on the fly

While navigating through the tree of already captured events in the Events pane, you may come across some events or scripts that you definitely do not want to trace. You can create a filter as described above but in this case you will have to leave the pane. With PhpStorm, you can create an exclusion rule based on any event or script, as soon as you have detected such event or script, right from the Events pane. The rule will be either added to the currently applied filter or a new filter will be created if the current setting is Capture All.

– To add an event to an exclusion filter on the fly, select the event to exclude and choose Mute <event name> event or Mute <script name> file.
  If a user-defined filter is currently applied, the new rule is added to it silently. If Capture All is currently active, the Spy-Js Capture Exclusions Dialog opens, where you can create a new filter based on the selected event or script or choose an existing filter and add the new rule to it.

Event Stack Pane

Once an event in the Events pane is clicked, its call stack is displayed in the Event Stack pane. The stack is represented by a tree of function calls. Each tree node represents the invoked function. Node text contains the total execution time, the script file name and the function name. When you click a node, the Quick Evaluation pane shows additional function call details, parameter values and return value, occurred exception details if there was one during the function execution.

The pane is synchronized with the editor, so you can navigate from an item in the stack tree to the corresponding trace file or source file.

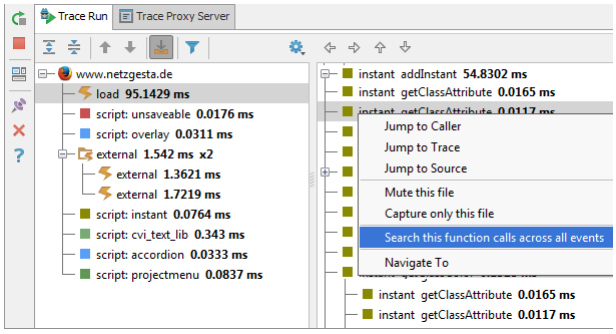
– A trace file is a write-protected prettified version of the script selected in the Events pane or the script whose function is double clicked in the Event Stack pane. A trace file is named `<file name>.js.trace`. When you double click an item in the stack tree or select it and choose Jump to Trace on the context menu of the selection, the corresponding trace file opens in the editor with the cursor positioned at the clicked function. Another approach is to press the Autoscroll to Trace toggle button and select various stack nodes. In this case, the trace file opens when you click an event or script in the Events pane.
  You can not only jump to a function but also to the place in the code where it was called from. To do that, select the required item and choose Jump to Caller on the context menu.

  The contents of the file are highlighted to display the code execution path of the selected stack node.
– When you are tracing an application with ECMASript6, CoffeeScript, and TypeScript code, Spy-js also generates mapped trace files. These are EcmaScript 6, TypeScript, or CoffeeScript trace files with the extensions `.ts.trace`, `.coffee.trace`, or `.js.trace`. The fragments of code in these files are highlighted as if they were really executed.
– You can also navigate to the source file displayed as is, without prettifying, by selecting an item in the Event Stack pane and choosing Jump to Source on the context menu of the selection. If the traced site is mapped with a PhpStorm project, PhpStorm detects the corresponding local file according to the mapping and opens this file in the editor. If you are tracing a site that is not mapped to a PhpStorm project, PhpStorm opens the read-only page source, just as if you chose View Page Source in the browser.
  When the traced site is mapped with a PhpStorm project, PhpStorm opens the source file on any attempt to edit the opened trace file.

Context Menu of a Script or Function

Item	Description
Jump to Caller	Choose this option to navigate to the fragment in the trace file from where the currently selected item was called. When you are tracing an application with ECMASript6, CoffeeScript, and TypeScript code, PhpStorm opens either the trace JavaScript file or the mapped trace file (TypeScript, CoffeeScript, or ECMAScript6): – If the Always open source mapped trace if available option is selected, the corresponding mapped trace file opens. – If the Always open source mapped trace if available option is not selected, the JavaScript trace file opens.
Jump to	Choose this option to navigate to the definition of the currently selected item in the trace file.

Trace	
Jump to Source	Choose this option to navigate to the definition of the currently selected item in the source file.
Mute this File	Choose this option to add the selected script to an exclusion filter on the fly, see Configuring the Range of Events to Capture .
Capture only this file	
Search this function calls across all events	Choose this option to navigate between the calls of a function within the whole trace (across all the traced events). This means that if you are tracing 5 pages in the browser and the Events pane, accordingly, shows 5 document nodes, PhpStorm searches for the calls of the selected function under all these nodes and displays the number of found calls of the function in the Status bar. The number of found calls is displayed in the Status bar, and the toolbar shows four previously hidden navigation chevron buttons.



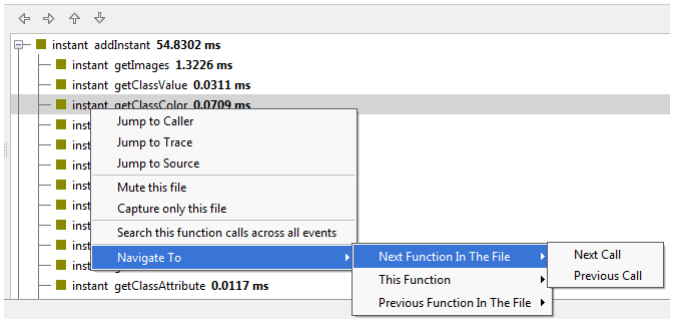
Use the chevron buttons to navigate within the found calls:

- To jump to the first detected call, click .
- To jump to the last detected call, click .
- To jump to the next detected call, click . The Status bar shows a message: `Occurrence <number> of <total number of detected calls>`
- To jump to the previous detected call, click .

The search results are reset and the search toolbar is hidden when you invoke another advanced search or navigation.

Also keep in mind that the number of call occurrences is calculated when you choose the Search this function calls across all events option. As you analyze the detected calls, the time passes, new events are captured, and the first detected call can happen to be already removed from the stack which means that it is no longer available for navigation.

Navigate to	Use the options under this item to move through the whole stack based on calls and locate the functions that have not been called, that is, locate the fragments of code that have not been executed and analyze the reason for them to be skipped. The following six actions are available: move to the next/previous call of the next/current/previous function in a trace file. The full list of actions is available from the context menu in the Event Stack pane. Moving to the next and previous calls of the selected function, to the previous call of the previous function, and to the next call of the next function are also available from the navigation toolbar of the Event Stack pane.



When you choose one of these actions, the cursor jumps to the call in the stack. If the Autoscroll to Trace toggle button is pressed, the corresponding trace file opens automatically with the cursor positioned at the call.

Synchronization between the Panes and the Editor

The Events and Event Stack panes are synchronized: when you click an event or script in the Events pane, its call stack is displayed in the Event Stack pane. To have also the corresponding trace file opened in the editor, press the Autoscroll to Trace toggle button on the toolbar.

The Event Stack pane is synchronized with the editor: when you click an item in the stack tree twice, the corresponding trace file opens in the editor with the cursor positioned at the clicked function.

To synchronize the Events pane directly with the editor, press the Autoscroll to Trace toggle button on the toolbar. In this case, as soon as you click a node in the Events pane, its call stack is displayed in the Event Stack pane and the corresponding trace file is opened in the editor. With the Autoscroll to Trace mode turned on, when you navigate through the Event Stack the corresponding files are also automatically opened in the editor with the corresponding functions highlighted.

Quick Evaluation Pane

When you click a node in the Event Stack pane, the Quick Evaluation pane shows additional function call details, parameter values and return value, occurred exception details if there was one during the function execution.

Context Menu of Function Call Details

The context menu is available from all items displayed in the pane.

Item	Description
Inspect	Choose this option to open the Inspect dialog box.
Copy Value	Choose this option to copy the value of the selected node to the clipboard.
Compare Value with	Choose this option to open the Differences viewer for files which displays the value of the selected node and the value in the clipboard so you can compare

Clipboard	them.
Copy Name	Choose this option to copy the name of the selected node to the clipboard.

Spy-js Capture Exclusions Dialog

The dialog box opens when you click the Capture Events button toolbar button in the Events pane of the Spy-Js Tool Window . In this dialog box, define custom filters to configure the range of events displayed and traced in the tool window. See Tracing with Spy-Js for details.

The dialog box consists of two panes. The left-hand, Exclusions, pane shows a list of already existing user-defined event filters. The right-hand, Exclusion, pane shows the details of the filter selected in the Exclusions pane.

Item	Description
Exclusions pane	The pane shows a list of all currently available user-defined filters. – To define a new event filter, click on the toolbar, then specify the filter name and exclusion rules in the Exclusion pane. – To create a filter based on an already existing one, make a copy of the source filter by selecting it and clicking , and then rename and edit the copy as required in the Exclusion pane. – To temporarily disable a filter, clear the check box next to it. – To remove a filter from the list, select it and click .
Exclusion pane	On this pane, configure custom event filters. For each filter, specify its name and create a list of exclusion rules. – To add a rule, click , the Add Condition to Exclusion dialog box opens. Type a pattern in the Value/pattern text box, in the Condition type drop-down list specify whether the pattern should be applied to event types or script names. Note, that glob pattern matching is used. When you click OK, PhpStorm brings you to the Spy-Js Capture Exclusions Dialog . – To edit a rule, select it in the list, click , and update the rule in the dialog box that opens. – To remove a rule, select it in the list and click .

Trace Proxy Server Tab

The tab consists of a toolbar and the console area that shows system information and error messages, information about the status of the Spy-js session, the availability of the proxy server, the currently active frames ("workers"), etc.

Trace Proxy Server Toolbar

Item	Tooltip and shortcut	Description
	Stop Trace Proxy Server	Click this button to terminate the currently running process externally by means of the standard `shutdown` script.

	Up/Down the Stack Trace
	`Ctrl+Alt+Up`
	`Ctrl+Alt+Down`
		Use these buttons to navigate through the stack trace from one reported error to another. These buttons are available only when errors are reported, for example, if the configuration file is corrupted.

	Use Soft Wraps	Click this button to toggle the soft wrap mode of the output.
	Scroll to the end	Click this button to navigate to the bottom of the stack trace and have the cursor jump to the corresponding location in the source code.
	Print	Click this button to send the console text to the default printer.
	Clear All	Click this button to remove all text from the console. This function is also available on the context menu of the console.

Structure Tool Window, File Structure Popup

Structure tool window

This tool window displays the structure of a file currently opened in the editor and having the focus, or selected in the Project tool window.

For diagrams, this tool window shows the diagram preview.

View | Tool Windows | Structure

`Alt+7`

File Structure pop-up window

This pop-up window displays the structure of a file, currently opened in the editor and having the focus.

Navigate | File Structure

`Ctrl+F12`

Both views help quickly navigate through the files' structure. Refer to the section Navigating with Structure Views .

This section describes the buttons on the title bar of the tool window and the options on the context menu of the title bar. Turn these options on and off to have elements of certain types hidden or shown and configure the way they are presented.

The buttons on the title bar are common for all language contexts. The set of options on the context menu depends on the context.

- Title Bar
- HTML, XML
- JavaScript, CoffeeScript
- PHP

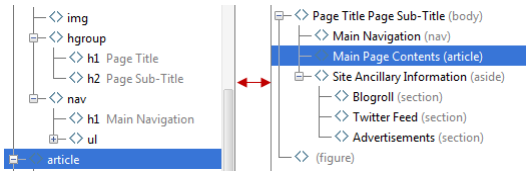
Tip Refer to the section Symbols for the member icons in the tree view of a file in the Structure tool window.

Title Bar

The buttons on the title bar are common for all language contexts.

Item	Tooltip	Description
	Collapse All `Ctrl+NumPad -`	Click this button to have all the nodes in the tool window collapsed.
	Expand All `Ctrl+NumPad Plus`	Click this button to have all the nodes in the tool window expanded.
		Click this button to open the context menu and configure the appearance of the tool window, its viewing mode , and the way it presents the structure of the current file by turning the menu items on or off.
	Hide `Shift+Escape`	Click this button to hide the tool window.

HTML, XML

Icon	Tooltip	Description
	Sort Alphabetically	Click this button to have the elements within a class sorted alphabetically.
	HTML5 Outline	Click this button on to view HTML 5 outline of a HTML file: 
	Collapse All `Ctrl+NumPad -`	Click this button to have all the nodes in the tool window collapsed.
	Expand All `Ctrl+NumPad Plus`	Click this button to have all the nodes in the tool window expanded.
	Autoscroll to Source	Click this button to enable automatic navigation to the line of source code that corresponds to the selected node when the focus switches to the editor.
	Autoscroll from Source	Click this button to have PhpStorm automatically move the focus in the Structure tool window to the node that corresponds to the code where the cursor is currently positioned in the editor.

JavaScript, CoffeeScript

Icon	Tooltip	Description
	Sort Alphabetically	Click this button to have the elements within a class sorted alphabetically.
	Group Methods by Defining Type	Click this button to have all the methods that override/implement the methods of a particular class/interface grouped under the node that corresponds to this class/interface.

	Show Fields	Click this button to have all fields (properties) shown in the tree.
	Show Inherited	Click this button to display all the methods and fields inherited by the current class and accessible from it. The inherited members are displayed gray to tell them from the members defined in the current class.
	Collapse All `Ctrl+NumPad -`	Click this button to have all the nodes in the tool window collapsed.
	Expand All `Ctrl+NumPad Plus`	Click this button to have all the nodes in the tool window expanded.
	Autoscroll to Source	Click this button to enable automatic navigation to the line of source code that corresponds to the selected node when the focus switches to the editor.
	Autoscroll from Source	Click this button to have PhpStorm automatically move the focus in the Structure tool window to the node that corresponds to the code where the cursor is currently positioned in the editor.

PHP

Title Bar

Besides the buttons that are common for all language contexts, the title bar in the PHP context contains two additional toggle buttons.

Icon	Tooltip	Description
PHP		Toggle this button to view the hierarchy of PHP elements. The button is always present on the toolbar but is enabled only when the current PHP class or file contains fragments of HTML code.
HTML		Toggle this button to view the hierarchy of HTML elements in a PHP class or file. The button is only available is the current PHP class or file contains fragments of HTML code.

Context Menu

Icon	Tooltip	Description
	Sort by Visibility	Click this button to have the items sorted by their visibility in the following order: public - protected - package local - private.
	Sort Alphabetically	Click this button to have the elements within a class sorted alphabetically.
	Show Inherited	Click this button to display all the methods and fields inherited by the current class and accessible from it. The inherited members are displayed gray to distinguish them from the members defined in the current class.
	Show Includes	Click this button to have all files included through `include` or `require` statements shown in the tree.
	Show Fields	Click this button to have all fields (properties) shown in the tree.
	Show Constants	Click this button to have constants shown in the tree.
	Collapse All `Ctrl+NumPad -`	Click this button to have all the nodes in the tool window collapsed.
	Expand All `Ctrl+NumPad Plus`	Click this button to have all the nodes in the tool window expanded.
	Autoscroll to Source	Click this button to enable automatic navigation to the line of source code that corresponds to the selected node when the focus switches to the editor.
	Autoscroll from Source	Click this button to have PhpStorm automatically move the focus in the Structure tool window to the node that corresponds to the code where the cursor is currently positioned in the editor.

Thumbnails Tool Window

Project tool window | context menu | Show Thumbnails

`Ctrl+Shift+T`

The Thumbnails tool window provides the functions similar to those of an image browser. It shows thumbnails for folders and image files, and lets you perform related navigation and image management tasks.

To open the tool window, use the Show Thumbnails command in the Project tool window (`Ctrl+Shift+T`). Then, unless you close the tool window, you can show and hide it as described in the section Manipulating the Tool Windows .

– Title bar context menu

– Title bar icons

– Toolbar icons

– Content pane: context menu commands

Title bar context menu

The title bar context menu provides the options for controlling the tool window viewing modes . It also contains the commands for associating the tool window with a different tool window bar , resizing and hiding the tool window.

To access the menu, right-click the window name (Thumbnails).

Note that most of the menu options may alternatively be accessed by means of the title bar icons .

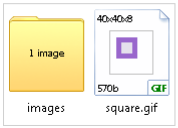
Item	Shortcut	Description
Pinned, Docked, Floating, Split Mode		These options let you control general appearance and behavior of the tool window, see Viewing Modes .
Move to		To associate the tool window with a different tool window bar , select this command, and then select the destination tool window bar (Top, Left, Bottom or Right).
Resize		To resize the tool window by moving one of its borders, select this command, and then select the necessary Stretch to option. Note that this command is not available in the floating mode.
Hide	`Shift+Escape`	Use this command to hide the tool window.

Title bar icons

Item	Shortcut	Description
		Use this icon to open the menu for changing the tool window viewing modes .
	`Shift+Escape`	Use this icon or shortcut to hide the tool window . When used in combination with the `Alt` key, clicking this icon hides all the tool windows attached to the same tool window bar .

Toolbar icons

The toolbar icons provide access to the most frequently used functions available in the tool window. The same and other functions are available as context menu commands in the content pane.

Item	Shortcut	Description
	`Backspace`	Use this icon or shortcut to move one level up in the folder hierarchy. The path to the current folder is shown on the title bar to the right of the tool window name.
	`Ctrl+numpad*`	Use this icon or shortcut to turn the Recursive option on or off. If this option is off, subfolders of the current folder and image files located in the root of the current folder are shown. If this option is on, the subfolders are not shown; the image files located in the current folder and all its subfolders are shown. The following picture shows the contents of the same folder with the Recursive off and on. Recursive: OFF  Recursive: ON
		Use this icon to show or hide a "chessboard". The "chessboard" (a checkered area) is shown underneath the images so that you can see transparent image areas for `.gif` and `.png` files. The following picture shows the same thumbnail with the "chessboard" shown and hidden.

✖	Ctrl+F4	Use this icon to close the tool window.

Content pane: context menu commands

When you right-click an item in the content pane, the context menu for this item is shown.

The following table lists and briefly explains the main context menu commands for thumbnails. Other commands, functionally, are similar to those in the Project tool window .

Item	Shortcut	Description
Browse	Enter	For a folder: use this command to view the folder contents (subfolders and images). The same command may alternatively be accessed by double-clicking a folder.
Jump to Source	F4 or Enter	For an image file: use this command to view the selected file in the editor. The same command may alternatively be accessed by double-clicking an image thumbnail.
Level up	Backspace	Use this command to move one level up in the folder hierarchy.
Recursive	Ctrl+numpad*	Use this command to turn the Recursive option on or off.
Show or Hide Chessboard		Use this command to show or hide the chessboard .
Close thumbnails	Ctrl+F4	Use this command to close the tool window.
Jump to External Editor	Ctrl+Alt+F4	For an image file: use this command to open the image in an external editor.

Time Tracking Tool Window

View | Tool Windows | Time Tracking

Item	Tooltip and Shortcut	Description
+	Open Task `Shift+Alt+N`	Create new task. Refer to the section Opening and Creating Tasks .
−	Remove Task	Delete selected tasks from the list.
✓	Show closed tasks	If this button is pressed, the list of tasks includes closed tasks. This button works as the second pressing keys `Shift+Alt+N`.
	Auto mode	If this button is pressed, time tracking automatically turns on for the current active task, and the button is disabled. If this button is not pressed, you have to start timer for the active task yourself.
/	Start/stop timer for the active task	Click this button to start or stop timer for the active task. When the timer is started, the button toggles to (Stop timer for active task). This button is disabled, when time tracking works in automatic mode (the button is pressed.)
	Post work item to bug tracker	Click this button send duration of work and optional comment to the bug tracker.
	Show time spent from last post of work item	

TODO Tool Window

View | Tool Windows | TODO

`Alt+6`

PhpStorm scans your project for comments in the source code that match the TODO patterns defined in the TODO dialog and displays results in the TODO tool window.

The TODO tool window consists of following tabs:

- Project tab that show the TODO items for the whole project.
- Current File tab.
- Scope Based tab that enables viewing TODO items pertaining to a certain scope, selected from the drop-down list, and ignoring the other items.
- Current Changelist, if version control is enabled.

This tool window helps you view, sort and group the TODO items in a convenient way, navigate to the source code and keep fixes under the version control.

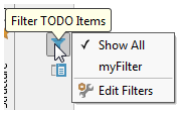
On this page:

- Toolbar buttons
- Context menu commands
- Title bar context menu and buttons

Toolbar buttons

Item	Tooltip and shortcut	Description
	Previous TODO `Ctrl+Alt+Up`	Navigate to the previous TODO item.
	Next TODO `Ctrl+Alt+Down`	Navigate to the next TODO item.
	Help `F1`	Use this icon or shortcut to open the corresponding help page.

	Expand all
	`Ctrl+NumPad Plus`
	Collapse all
	`Ctrl+NumPad -`
		Use these buttons to have all nodes expanded or collapsed.

	Autoscroll to Source	Toggle the Autoscroll to source mode. When this button is pressed, every time the node is focused, the corresponding line of source code is highlighted in the editor.
	Filter TODO items	Click this button to select the desired filter from the list, or invoke the TODO dialog and edit the list of TODO patterns and filters as required. 
	Preview Usages	If this button is pressed, a pane to the right shows the source code of the selected file, with the corresponding TODO item highlighted.
Tip The following buttons are available in the Project and Scope Based tabs.		
	`Ctrl+D`	If this button is pressed, the TODO items show under the corresponding module or library node.
	`Ctrl+P`	If this button is pressed, the TODO items show under the corresponding packages.
	`Ctrl+F`	If this button is pressed, the TODO items show as a flat list. Thus, if a package is somewhere deep within your project, you do not need to dig deep into the hierarchy.

Context menu commands

Item	Keyboard Shortcut	Description
Jump to Source	`F4`	Navigate to the selected usage in the source code.
Local History		Show Local History submenu for the selected search result. Refer to the Local Version Control procedures for details.
<VCS>		Show menu of the VCS, associated with the directory. See Version Control Procedures and Reference for details.

Title bar context menu and buttons

The title bar context menu provides the options for controlling the tool window viewing modes . It also contains the commands for associating the tool window with a different

tool window bar , resizing and hiding the tool window.

To access the menu, right-click the window title bar.

Note that most of the menu options may alternatively be accessed by means of the title bar buttons.

Toolbar icon	Context menu command	Description
	Pinned, Docked, Floating, Split Mode	These options let you control general appearance and behavior of the tool window, see Viewing Modes .
	Move to	To associate the tool window with a different tool window bar , select this command, and then select the destination tool window bar (Top, Left, Bottom or Right).
	Resize	To resize the tool window by moving one of its borders, select this command, and then select the necessary Stretch to option. Note that this command is not available in the floating mode.
Shift+Escape	Hide	Use this command to hide the tool window. Tip When used in combination with the Alt key, clicking this button hides all the tool windows attached to the same tool window bar .

TypeScript Compiler Tool Window

On this page:

- Current Errors Pane
- Project Errors Pane
- Toolbar
- Context Menu
- Console

Current Errors Pane

The pane shows a list of all the discrepancies detected in the file which is opened in the active editor tab. At the top the full path to the file is displayed. For each discrepancy, PhpStorm provides a brief description and information about the line number where the error occurred. In addition to this you can navigate to the code in question right from the corresponding error message by choosing Jump to Source on the context menu.

Project Errors Pane

The pane shows a list of all the discrepancies detected in all the files in the current project after you run the built-in compiler against the entire project by clicking on the toolbar. The error messages are grouped by files in which they were detected. For each discrepancy, PhpStorm provides a brief description and information about the line number where the error occurred. In addition to this you can navigate to the code in question right from the corresponding error message by choosing Jump to Source on the context menu.

Toolbar

The toolbar is common for the Current Errors and the Project Errors panes.

Item	Tooltip and Shortcut	Description
	Compile Current File	Click this button to run the compiler against the entire file opened in the active editor tab. This approach is helpful if you do not want the compiler to wake up on any change as you edit your code and for this purpose the Track Changes check box on the TypeScript page of the Settings dialog box is cleared. If you click this button from the Project Errors pane, after compilation PhpStorm switches to the Current Errors pane.
	Compile All	Click this button to run the compiler against all the TypeScript files in the current project. If you click this button from the Current Errors pane, after compilation PhpStorm switches to the Project Errors pane.
	Close `Ctrl+Shift+F4`	Click this button to terminate the compiler and close the tool window.

	Expand all
	`Ctrl+NumPad Plus`
	Collapse all
	`Ctrl+NumPad -`
		Use these buttons to have all nodes expanded or collapsed.

	Clear All	Click this button to remove all the error messages from the currently active pane.

Context Menu

The context menu is common for the Current Errors and the Project Errors panes.

Item	Description
Jump to source	Choose this option to open the file where the selected error occurred during compilation and navigate to the fragment of code which caused the error.
Copy	Choose this option to copy the selected error message with the information on the file, the line, and the column where the error occurred during compilation.

Console

The tab shows the log of the built-in compiler since the tool window was opened.

Version Control Tool Window

View | Tool Windows | Version Control

This tool window is available if version control is enabled for your project.

The tool window accomodates several viewes that display VCS-realted information and allow you to manage changelists, perform VCS-specific actions, view changes made by other team members, etc.

The tool window consists of several tabs:

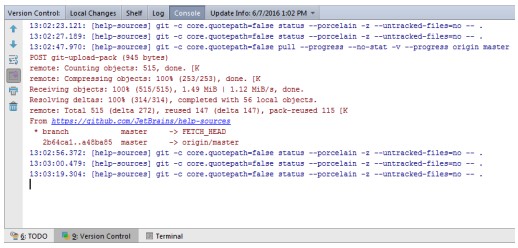
- Console tab: this tab shows the results of executing VCS-related commands.
- Local Changes tab: this tab is always present and shows the list of files that have been modified locally and have not been committed to the repository yet.
- History tab: this tab is added to the Version Control tool window when the Show History command is invoked through VCS | <specific_VCS>.
- Integrate to Branch Info View tab: this view is available after running integration with the Run status after update setting specified.
- Log tab: this tab is only available if you are using Git or Mercurial as your version control system. It shows all changes committed to all branches of the local and remote repositories, or to a specific branch or repository.
- Repository and Incoming tabs: the Repository tab shows the changes committed to the repository under the VCS roots within the current project. The Incoming tab shows the changes commited to the repository by other team members, and not checked out locally.
- Shelf tab: this tab is added to the tool window when you shelve a change or a changelist.
- Update Info tab: this tab becomes available when local information is synchronized with the server.

Console Tab

View | Tool Windows | Version Control | Console

The tab displays:

- Version control-related commands generated based on the settings you specify through the PhpStorm interface.
- The results of executing version control-related commands.



Toolbar

Item	Tooltip and Shortcut	Description

	Up the stack trace / Down the stack trace
	`Ctrl+Alt+Up`
	`Ctrl+Alt+Down`
		Click these buttons to navigate up or down in the stack trace and have the cursor jump to the corresponding location.

	Use Soft Wraps	Click this button to toggle the soft wrap mode of the output.
	Scroll to the end	Click this button to navigate to the bottom of the stack trace and have the cursor jump to the corresponding location.
	Print	Click this button to send the console text to the default printer.
	Clear All	Click this button to remove all text from the console. This function is also available on the context menu of the console.

Context Menu Options

Item	Description
Compare with Clipboard	Opens the Clipboard vs Editor dialog box that allows you to view the differences between the selection from the editor and the current clipboard content. This dialog is a regular comparing tool that enables you to copy the line at caret to the clipboard, find text, navigate between differences and manage white spaces.
Copy URL	Choose this command to copy the current URL to the system clipboard. This command only shows on a URL, if it is included in an application's output.
Create Gist	Choose this command to open the Create Gist dialog box.
Clear All	Clears the output window.

History Tab

The History tab is added to the Version Control tool window on invoking the Show History command for a file or directory through the menu of a particular VCS. A new tab is created for each file or directory, with the name File <name> History. The set of toolbar buttons differs slightly depending on your version control systems.

All commands available from the toolbar are also available from the context menu of a selection.

Item	Tooltip and Shortcut	Description
	Compare Ctrl+D	Click this button to compare the selected revision of a file with its previous revision in the Differences viewer for files .
	Compare with Local	Click this button to compare the selected revision of a file with its local copy in the Differences viewer for files .
	Compare all classes from revision on UML Ctrl+Shift+D	Click this button to view all classes of the selected revision as a UML Class diagram. See section Viewing Changes as Diagram .
	Create Patch	Click this button to create a patch from the selected revision.
	Get	Click this button to retrieve the selected revision. If the local copy has already been modified, PhpStorm prompts to overwrite the local version, or cancel the operation.
	Annotate	Click this button to open the selected revision of a file in the editor with annotations.
	Show All Affected Files Shift+Alt+A	Click this button to open the Paths Affected in Revision dialog where you can view all files that were modified in the selected revision.
	Copy Revision Number	Click this button to copy the revision number of the commit that the selected file belongs to to the clipboard.
	Show Branches	Click this button to show branches.
	Show All Revisions Submitted In Selected Changelist	Click this button to display the list of all revisions committed in the same changelist as the selected revision of a file.
	Refresh	Click this button to refresh the current information.
	Show Details	Click this button to show the commit message for the selected revision.
	Close Ctrl+Shift+F4	Click this button to close the current history tab.
	Help F1	Click this button to show reference information.

File revisions are shown in regular font. The revision that currently exists in your working copy, is shown in bold font.

Integrate to Branch Info View

View | Tool Windows | Version Control

This view is available after running integration with the Run status after update setting specified. The view displays a list of files or packages affected by the latest integration. The items are displayed under the following nodes:

- Modified
- Merged
- Not in repository
- Locally added

Item	Tooltip and shortcut	Description
		Click this button, to group information within nodes by packages. If the button is released, files are presented in plain lists.
	Ctrl+NumPad Plus	Click this button to expand all nodes.
	Ctrl+NumPad -	Click this button to collapse all nodes.
		Click this button to close the view.
	F1	Click this button to show the corresponding reference page.

Local Changes Tab

VCS | Show Changes View - Local Changes

View | Tool Windows | Version Control - Local Changes

`Alt+9`

The Local Changes tab shows the files that have been modified locally and have not yet been committed to the repository.

> **Note** You can assign a custom shortcut for the Show Local Changes action in Settings | Keymap | Version Control Systems to open the Local Changes tab.

Use this tab to commit and revert changes, manage changelists , view differences , view changes in UML Class diagram , and clean up locked folders.

In this topic:

- Toolbar
- Changelists pane
  - Context menu of a selection
- Preview Diff Pane

Toolbar

Item	Tooltip and Shortcut	Description
	Refresh `Ctrl+F5`	Click this button to refresh the status of all files in your workspace, both made through PhpStorm or through any other application. Note that if you are using Perforce as your version control system, only the status of files modified through PhpStorm will be updated. This approach improves performance, as it does not require connecting to the server, but it does not let you get an update on the changes made outside PhpStorm, for example through the p4v client application. If you want to get an update on all changes to your workspace, use the Force Refresh option.
	Force Refresh	This button is only available if you are using Perforce as your version control system. Click this button to refresh the status of files in your workspace, both modified through PhpStorm or through other applications.
	Commit Changes	Click this button to check in the selected change or changelist. You can also attach and detach Perforce jobs to/from changelists via the Commit Changes dialog.
	Revert `Ctrl+Alt+Z`	Click this button to roll back the selected changes.
	New Changelist `Alt+Insert`	Click this button to create a new changelist .
	Delete Changelist `Delete`	Click this button to delete the selected changelist. Note that you cannot delete the default changelist.
	Set Active Changelist	Click this button to make the selected changelist active . The active changelist is highlighted.
	Move to Another Changelist `F6`	Click this button to move the selected file to another changelist.
	Show Diff `Ctrl+D`	Click this button to view the differences between your local version of the selected file and its latest version in the repository.
	Show Changes `Ctrl+Shift+Alt+D`	Use this button to show classes from the selected changelist in a UML Class diagram (for details see Viewing Changes as Diagram).

	`Ctrl+NumPad Plus`
	
	`Ctrl+NumPad -`
		Click these buttons to expand or collapse all nodes.

	Group by Directory `Ctrl+P`	Click this button to display the changed files grouped by directories. If the button is released, the changed files are grouped by changelists.
	Copy `Ctrl+C`	Click this button to copy the path to the selected file to the clipboard.
	Show Ignored Files	Click this button to show the Ignored files node with the list of existing files ignored by the VCS.
	Configure Ignored Files	Click this button to configure the list of files that will be ignored by your version control system.
	Preview Diff	Click this button to have PhpStorm open or close the Preview Diff pane to compare the current file with the latest committed revision.
	`F1`	Click this button to show the corresponding PhpStorm help page.

Changelists pane

This pane shows all your changelists, and the files that have been modified in each changelist.

If new files have been added to your project that have not yet been added to a version control system, the Unversioned Files node appears under which all such files are listed. If you have a large number of unversioned files (over 50), they are not displayed in the changelists pane. Instead, the Click to browse link appears. Click this link to open the Unversioned Files dialog to see the list of files that have not been added to a version control system. You can quickly delete unversioned files from the Changelists pane or the Unversioned Files dialog by pressing `Delete`, or add them to the VCS by pressing `Ctrl+Alt+A`.

Context menu of a selection

Item	Shortcut	Description
Commit Changes	N/A	Select this option to check in the selected file or changelist. You can also attach and detach Perforce jobs to changelists via the Commit Changes dialog.
Revert	`Ctrl+Alt+Z`	Select this option to roll back the selected changes.
Move to Another Changelist	`F6`	Select this option to move the selected item to another changelist.
Show Diff	`Ctrl+D`	Select this option to view the differences between your local copy and the latest version in the repository.
Jump to Source	`F4`	Select this option to open the selected file(s) in the editor.
New Changelist	`Alt+Insert`	This option is only available if a changelist is selected, not single files. Select this option to create a new changelist .
Delete	N/A	This option is only available if single files are selected, not a changelist.
Check Out	N/A	This option is only available if a file under the Modified without Checkout node is selected. Use this option to check out the selected file from the repository.
Add to VCS	N/A	This option is only available if a file under the Unversioned Files node is selected. Use this option to add the selected files to your version control system.
Ignore	N/A	This option is only available if a file under the Unversioned Files node is selected. Use this option to ignore the selected file if you want to leave it unversioned.
Create Patch	N/A	Select this option to create a patch .
Shelve Changes	N/A	Select this option to shelve the selected changes .
Shelve in Perforce	N/A	This option is only available if you are using Perforce as a version control system. Select this option to shelve your changes in Perforce. You will be asked to select which files you want to shelve and provide a description. After you've shelved your changes, the corresponding changelist will appear. You can unshelve it any time from the changelist's context menu.
Refresh	`Ctrl+F5`	Select this option to refresh the status of files in your workspace.
Local History	N/A	Select this option and choose one of the following from the popup menu: – Show History: select this option to View local history of the selected file or folder. – Put Label: select this option to add a lable to the current version of the selected file or folder.
<Specific version control system>	N/A	Select this option to invoke a popup menu with options specific for the version control system you are using.

Preview Diff Pane

This pane opens when you click the Preview Diff button on the toolbar. In this pane you can examine the changes made to the selected file compared to its base revision.

The pane consists of two areas:

– The affected code as it was in the base revision.

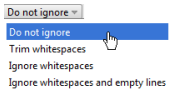
– The affected code as it is after a change has been made.

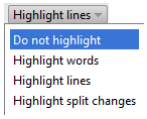
To close the pane, click the Preview Diff button once again.

Item	Tooltip and Shortcut	Description
	Previous Difference / Next Difference `Shift+F7` `F7`	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to click the arrow buttons `F7` / `Shift+F7` once more and compare other files, depending on the Go to the next file after reaching last change option in the Differences Viewer settings . This behavior is supported only when the Differences Viewer is invoked from the Version Control tool window.

	Compare Previous/Next File
	
	`Alt+Left`
	`Alt+Right`
		Click these buttons to compare the local copy of the previous/next file with its update from the server.
		Tip These buttons are available only when the Differences Viewer is invoked from the Version Control tool window.

	Jump to Source `F4`	Click this button to open the selected file in the active pane in the editor. The caret will be placed in the same position as in the Differences Viewer.
Viewer type		Use this drop-down list to choose the desired viewer type. The side-by-side viewer has two panels; the unified viewer has one panel only. Both types of viewers enable you to – Edit, for example, type text. Note that one can change text only in the right-hand part of the default viewer, or, in case of the unified viewer, in the lower ("after") line. – Perform the Apply/Append/Revert actions.

Whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

Highlighting mode		Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted – Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Collapse unchanged fragments	Click this button to collapse all unchanged fragments in both files. The amount of non-collapsible unchanged lines is configurable in the Diff & Merge settings page.
	Synchronize scrolling	Click this button to simultaneously scroll both differences panes; if this button is released, each of the panes can be scrolled independently.
	Editor settings	Click this button to reveal the drop-down list of available check commands. Select or clear these check commands to show or hide line numbers, indentation guides, white spaces, and soft wraps. Note that same commands are duplicated in the context menu of the differences viewer gutter. Right-click the gutter to show the menu of check commands.
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools settings page. This button only appears on the toolbar when the Use external diff tool option is enabled in the External Diff Tools settings page.
	Help F1	Click this button to show the corresponding help page.

The most useful shortcuts in the Diff Pane are the following:

Shortcut	Description
Ctrl+Shift+D	Use this keyboard shortcut to show the popup menu of the most commonly user diff commands.
Ctrl+Tab	Use this keyboard shortcut to switch between the left and the right panes.
Ctrl+Shift+Tab	Use this keyboard shortcut to select the position obtained by Ctrl+Tab in the opposite pane.
Ctrl+Z / Ctrl+Shift+Z	Use this keyboard shortcut to undo/redo a merge operation. Conflicts will be kept in sync with the text.

Log Tab

VCS | Show Changes View - Log

View | Tool Windows | Version Control - Log

`Alt+9`

The tab is only available if your project is integrated with Git or Mercurial .

This tab shows all changes committed to all branches of the local and remote repositories, or to a specific branch or repository.

Note that the log is updated automatically in the background mode on every change (a commit, fetch, rebase, etc.), even if the Log tab is closed.

Note You can assign a custom shortcut for the Show VCS Log action in Settings | Keymap | Version Control Systems to open the Log tab.

The tab contains the following panes:

- The Commits pane is located on the left of the tool window and shows the commits to all or selected branches from the local and remote repositories.
- The Changed Files pane is located on the right of the tool window and shows the list of files that were modified and committed within the selected commit.

Commits Pane

The commits pane consists of three areas:

- Commits
- Commit Details
- Toolbar . Some of the toolbar commands are duplicated in the context menu .

Commits

This area shows a list of all commits performed to the selected branch, or to all branches. For each commit, the list shows the commit message, the author, and the commit date. The latest commit in each branch is supplied with a color label with the name of the branch in which it was performed.

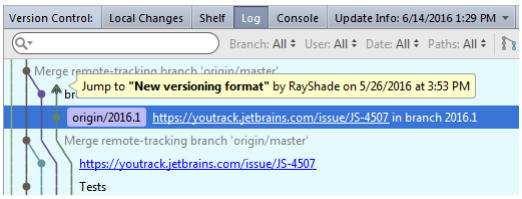
There are the following labels:

- local (Mercurial) and regular (Git/Mercurial) for tags.
- tip (Mercurial) for the latest revision in the repository.
- HEAD (Git/Mercurial) for the current working revision.

For Git, the color of the label depends on the branch type (local or remote).

For Mercurial, there are different color labels for bookmarks, open heavy branches and closed heavy branches.

Note that clicking an arrow takes you to the next commit in a long branch:



In multi-root projects, the colored stripe on the left indicates which root the selected commit belongs to (each root is marked with its own color). Hover the mouse cursor over the colored stripe to invoke a tip that shows the root path. You can also enable the Show Root Names option if you want to expand the Roots column with full root names.

Committed changelists often correspond to issues in tracking systems. You can jump to such issues in a browser right from the Commits pane. This functionality is available if:

- The pattern of the bug tracking system is specified in the Issue Navigation Settings dialog box.
- The corresponding issue number is mentioned in the commit message.

After issue navigation has been configured, issue numbers in commit messages are rendered as links. Clicking such link brings you to the corresponding page of your issue tracker.

Commit Details

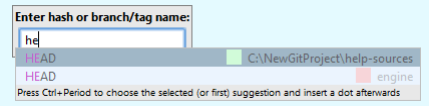
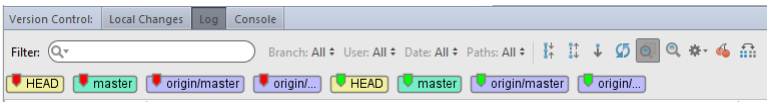
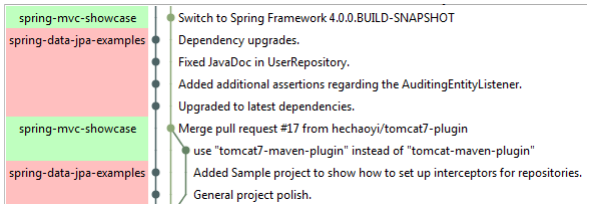
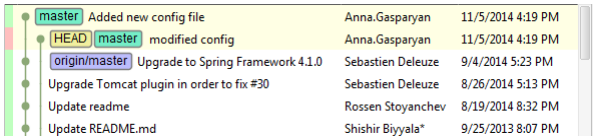
This area is displayed when the Show Details option is enabled.

This area shows the details on the commit selected in the Commits list, such as date, time, author, root and branches.

If the selected commit is contained in more than six branches, only the first six are displayed and the Show All link appears that you can click to expand a complete list of branches.

Toolbar

Item	Tooltip and Shortcut	Description
Filter	N/A	Use this text box to search through the list of commits. You can enter full or parts of commit names or messages, revision numbers, or regular expressions. As you type, the commits containing your search string are displayed. To finalize the search, press `Enter`. You can quickly switch the focus to the search field by pressing `Ctrl+L`.

	N/A	Click this button to show previously used search patterns.
	N/A	Click this button to clear the search history list.
Branch	N/A	Use this drop-down list to filter the commits by the branch. If you want to see commits from all local and remote branches, select All.
User	N/A	Use this drop-down list to filter the commits by the author. To view the commits of a specific author, click Select and start typing the author's name. To view commits by all users, select All.
Date	N/A	Use this drop-down list to filter the commits by a time-frame or a specific date. To view the commits made on a specific date, click Select and specify the date. To view commits made on all dates, select All.
Paths	N/A	Use this drop-down list to filter the commits by the folder (for projects that have one root), or by the root and folder (for multi-root projects). To view the commits to a specific folder, click Select Folders and specify the folder name. For multi-root projects, you can also select the check-box next to one or several roots in the Roots section.
	IntelliSort	If this option is enabled, you get a more convenient way to view merges by displaying the incoming commits first, directly below the merge commit.
	Collapse linear branches	Click this button to collapse all branches on the graph so that a dotted line is shown instead of successive commits. It is also possible to collapse an individual expanded branch by clicking it.
	Expand all branches	Click this button to expand all collapsed branches to show successive commits on the graph. It is also possible to expand an individual collapsed branch by clicking it.
	Show long edges	If this option is enabled, long branches are displayed in full, even if there are no commits in them. If this option is disabled (by default), long branches are replaced with a down arrow.
	Refresh Ctrl+F5	Click this button to refresh the information in the Commits pane.
	Show details	Click this button to show the Commit details pane .
	Go to Hash/Branch/Tag Ctrl+F	Click this button and specify a hash, tag or branch you want to jump to in the text field.  You can select a reference with the same name from different repositories. The name of each repository is displayed on the right side of the list along with its color indicator.
	Quick settings	Click this button to invoke one of the following commands: – Show Branches Panel: enable this option if you want a panel listing all branches to be displayed:  You can click on a label in the branches panel to jump to the corresponding branch in the Commits list. – Show Root Names: enable this option if you want to expand the Roots column on the left showing full root names. Note that this option is only available for multi-root projects.  – Highlight My Commits: enable this option if you want your commits to be highlighted with the yellow background in the commits list. 
	Cherry-pick (for Git) Graft (for Mercurial)	Click this button to apply the changes from the selected commit to the current branch .
	Highlight non-picked commits	Note that this command is only available if you are using Git as your version control system. Click this button to highlight the commits from the selected branch that have not yet been applied to the current branch.

Context menu commands

Item	Description	Available in
Copy Revision Number	Use this command to copy the revision number of the selected commit to the clipboard.	Git Mercurial
Create Patch	Use this command to create a patch based on the selected commit.	Git Mercurial
Cherry-pick (Git)	Use this command to apply the changes from the selected commit to the current	Git

	branch .	
Graft (Mercurial)		Mercurial
Checkout Revision	Use this command to check out the state of files recorded in the selected commit.	Git
Update to Revision	Use this command to change your working copy parent revision to the selected commit. New commits will carry on from the revision (commit) you update to.	Mercurial
New Branch	Use this command to create a new branch based on the selected commit.	Git Mercurial
New Tag	Use this command to add a new tag to the selected commit.	Git Mercurial
Branch <branch_name> / Branches	This command appears for all branches that point to the selected commit (Branch <branch_name> if there is one branch, or Branches if there are multiple branches) and provides the same options as the ones available from the Branches popup and submenu : – Checkout as new local branch – Compare – Rebase onto – Merge – Delete If the Control repositories synchronously option is enabled, and the selected branch exists in multiple repositories, an additional menu option named In All Repositories appears that allows you to perform the same operations in all repositories simultaneously.	
Reset Current Branch to Here	Use the command to reset the current branch head to the selected commit. In the Git Reset dialog that opens, select the mode in which the working tree will be updated.	Git
Open in GitHub	Use this command to open the page that corresponds to the selected commit on GitHub .	Git
MQ	Use this submenu to manage Mercurial Queues : – Import: use this command to turn the selected changeset into a patch. – Goto patch: use this command to open the MQ: <project_name> tab that shows a queue of patches that have not been applied yet. – Rename Patch: use this command to rename the selected patch. – Finish Patches: use this command to turn the selected patch into a permanent changeset.	Mercurial

Changed files pane

This pane shows a list of files that were modified within the currently selected commit.

Toolbar

Many of the options available in the toolbar are duplicated in the context menu.

Item	Tooltip and Shortcut	Description
	Show Diff Ctrl+D	Click this button to open the Differences viewer for files where you can compare the current and the previous revision of the selected file.
	Show Diff with Local	Click this button to show the differences between the selected revision of the selected file and its current local copy.
	Edit Source F4	Click this button to open the local copy of the selected file for editing.
	Open Repository Version	Click this button to open the repository version of the selected file for editing.
	Revert Selected Changes	Click this button to roll back the changes in the selected file.
	Group by Directory Ctrl+P	Click this button to transform a flat list of files into a tree of packages with files.
	Expand All/Collapse All Ctrl+NumPad Plus	Click this button to expand/collapse all nodes. Note that these buttons are only available only when tree-view is enabled.

Context menu commands

Item	Tooltip and Shortcut	Description
	Show Diff Ctrl+D	Click this button to open the Differences viewer for files where you can compare the current and the previous revision of the selected file.
	Show Diff with Local	Click this button to show the differences between the selected revision of the selected file and its current local copy.
	Edit Source F4	Click this button to open the local copy of the selected file for editing.
	Open Repository	Click this button to open the repository version of the selected file for editing.

	Version	
	Revert Selected Changes	Click this button to roll back the changes in the selected file.

Repository and Incoming Tabs

VCS | Show Changes View - Repository/Incoming

View | Tool Windows | Version Control - Repository/Incoming

Alt+9

The Repository and Incoming tabs are only available for non-distributed version control systems (i.e. all VCSs supported by PhpStorm except for Git and Mercurial).

The Repository tab shows the changes committed to the repository under the VCS roots within the current project. The Incoming tab shows the changes committed to the repository by other team members, and not yet checked out locally. Both tabs display the information stored in the history cache. The number of changelists displayed depends on the cache scope .

Each tab contains the following panes:

- The Changelists pane shows changelists.
- The Changed Files pane shows the list of files that were modified and committed within the selected changelist.

Note that if you are using SVN 1.5 or higher both on the server and in the local working copies, the Repository tab also features a Merge Info pane that configures the view in the other two panes and provides control over integration between branches.

Changelists Pane

The pane shows the changelists committed and stored in the history cache. When you click a changelist, the files affected by the selected commit are displayed in the Changed Files pane.

Committed changelists often correspond to issues in tracking systems. You can have such issues opened in the browser right from the Changelists pane. This functionality has the following prerequisites:

- The pattern of the bug tracking system is specified in the Issue Navigation Settings dialog box.
- The corresponding issue number is mentioned in the commit message.

After issue navigation has been configured, issue numbers in commit messages are rendered as links. Clicking such link brings you to the corresponding page of your issue tracker.

Item	Tooltip and Shortcut	Description	Available In
	Refresh Ctrl+F5	Click this button to refresh the information in the view.	Both tabs
	Show Details Ctrl+Q	Click this button to show the following information on the selected changelist: – Changelist number – User and client name – Date and time of commit	Both tabs
	Create Patch	Click this button to create a patch based on the selected changelist.	Repository tab
	Revert Changes	Click this button to create a *reverse patch* for the selected changelist and roll back the changes made previously. You can use this action to revert changes committed by any user. The Select Target Changelist dialog box opens. Note that if the reverse patch applies to a version committed earlier, this rollback attempt may fail because of the conflicts with the later changes.	Repository tab
	Clear	Click this button to clear the history cache. The commits list will be emptied. To restore it, click Refresh .	Repository
	Edit Revision Comment	Click this button to edit the message for the selected commit.	Repository
	Update Project Ctrl+T	Click this button to update the project to the latest available version.	Incoming tab
	Expand All Ctrl+NumPad Plus	Click this button to expand all nodes.	Both tabs
	Collapse All Ctrl+NumPad -	Click this button to collaps all nodes.	Both tabs
	Copy Ctrl+C	Click this button to copy the commit message of the selected changelist to the Clipboard.	Both tabs
	Help F1	Click this button to show the corresponding help topic.	Both tabs
	Highlight Integrated	Click this button to have the Merge Info pane displayed. The button is enabled only when both the server side and the client side use Subversion 1.5.	Repository tab

Filter by	Use this drop-down list to hide the changelists that are of no interest to you, and only view only the changelists that satisfy a certain criterion. The following options are available: – User: select this option to filter the commits by the user. – Structure: select this option to filter the commits by the target module or folder. – Client: select this option to filter the commits by the computer from which they were made. – None: select this option to turn off filtering and return to the default view.	Both tabs
Group by	Use this drop-down list to group changelists following a certain criterion. The following options are available: – Date: select this option to group the commits by date. – User: select this option to group the commits by users. – Client: select this optio to group the commits by the computer from which they were made.	Both tabs
Search	Use this text box to enter a search pattern and locate the commits whose commit messages matches the specified string. As you type, the list dynamically reduces to show the changelists with the commit messages that match the specified pattern. To save the search pattern, press Enter. To view the list of recent search patterns, click the button. To clear the list of search patterns, click the button.	Repository tab

Changed files pane

Item	Tooltip and Shortcut	Description
	Show Diff Ctrl+D	Click this button to show the differences between the current and the previous revision of the selected file.
	Show Diff with Local	Click this button to show the differences between the selected revision of the selected file and its current local copy.
	Edit Source F4	Click this button to open the local copy of the selected file for editing.
	Open Repository Version	Click this button to open the repository version of the selected file.
	Revert Selected Changes	Click this button to revert the changes to the selected file and roll back to its previous revision.
	Integrate to Branch	Click this button to integrate the changes from the selected file to the target branch.
	Compare Subversion Properties	This option is only available if you are using Subversion as your version control system. Click this button to view the differences in file properties between the current version and the previous revision.
	Show History	Click this button to open the History view of the selected file in the Version Control tool window.
	Group by Directory Ctrl+P	Click this button to transform a flat list of files into a tree of packages with files.
	Expand All Ctrl+NumPad Plus	Click this button to expand all nodes. The button is available only when the files in the pane are displayed grouped by directories.
	Collapse All Ctrl+NumPad -	Click this button to collapse all nodes. The button is available only when the files in the pane are displayed grouped by directories.

Merge Info pane

The pane is available only if you are using SVN 1.5 or higher both on the server and in the local working copies.

In this pane, specify a pair of branches whose integration with each other you want to monitor. The Changelists pane will show the changelists related to the specified branches and provide the information on the integration status of each changelist.

You can specify several pairs of branches if several projects or roots are involved.

Item	Tooltip and Shortcut	Description
From		Specify the URL address of the source branch. PhpStorm suggests the URL address selected in the Checkout from Subversion dialog box.
To		Do the following: – Specify the path to the target branch.

		Click ... or press `Shift+Enter` to open the Select Branch dialog box. – Specify the path to the local working copy to which you will apply patches created based on the selected changelists. Click ... to open the Configure Working Copy Paths dialog box and select a working copy.
	Highlight Integrated	Click this button to have each changelist in the Changelists pane supplied with an indication of whether it is integrated or not.
	Integrate To Branch	Click this button to integrate the selected changelist into the working copy. The Integrate To Branch dialog box opens.
	Undo Integrate To Branch	Click this button to revert the last integration of the selected changelist into the working copy.
	Mark As Merged	Click this button to indicate that the selected changelist is integrated into the working copy without actually integrating the changelist. The action affects the administrative information in the `.svn` folder. The icon next to the selected changelist changes from to .
	Mark As Not Merged	Click this button to indicate that the selected changelist is not integrated into the working copy without actually reverting integration. Update the administrative information in the `.svn` folder. The icon next to the selected changelist changes from to .
	Filter Out Integrated	Click this button to display only changelists that have not been integrated into the working copy.
	Filter Out Not Integrated	Click this button to display only changelists that have been integrated into the working copy.
	Filter Out Others	Click this button to hide extraneous changelists in the Changelists pane. Extraneous changelists are changelists that are managed in another VCS or are located under another root.
	Show Working Copies	Click this button to open the Subversion Working Copies Information dialog box.
	Refresh	Click this button to refresh the information in the Changelists pane.

Shelf Tab

View | Tool Windows | Version Control - Shelf

N/A

The tab is added to the Version Control tool window when you shelve a change or a changelist, and is displayed until you permanently remove all shelved changes, including the already unshelved ones, and imported external patches.

By default, this tab shows all shelved changes that have not been unshelved yet. Changes are grouped into shelves. A shelf is a changelist created when you shelve changes. A shelf is identified by the commit message. You can have PhpStorm show the unshelved changes. They can be restored and re-applied as many times as necessary, until they are removed permanently.

For details, see Shelving and Unshelving Changes .

Note You can assign a custom shortcut for the Show Shelf action in Settings | Keymap | Version Control Systems to open the Shelf tab.

Toolbar

Item	Tooltip and shortcut	Description
	Show Diff Ctrl+D	Choose this option to open the Differences viewer for files and compare the shelved version of a file with its current local version.
	Create Patch	Choose this option to create a patch file based on a shelved change. In the Create Patch dialog box that opens, specify the file to save the patch in, and the changes to create a patch from. By default, all changes from the selected shelf are involved in the patch creation. To view which changes are included, click the Selected link. For details, see Creating Patches .
	Show/Hide already unshelved	Click this button to have PhpStorm show or hide all available shelved changes, both already applied and not. Note that this button is duplicated on the context menu.
		Click this button the reveal the menu of actions described below .

Context menu

The context menu and the toolbar button provide the same set of actions. The context menu is available by right-clicking a change, a change list, or anywhere in the tab.

Item	Shortcut	Description
Unshelve Changes		Choose this option to apply all changes from the selected shelf. In the Unshelve Changes to Changelist dialog box box that opens, specify the changelist you want to add the changes to. For details, see Unshelving Changes .
Unshelve	Ctrl+Shift+U	Choose this option to select specific changes you want to unshelve instead of applying an entire shelf. In the Unshelve Changes to Changelist dialog box box that opens, specify the changes you want to apply, and the changelist you want to add these changes to. For details, see Unshelving Changes .
Restore		Choose this option to re-activate the unshelved changes. By default, unshelved changes are no longer shown in the list of available shelved changes, therefore you first need to have PhpStorm display it by choosing Show Already Unshelved. You can restore any change as many times as you need until the change is permanently removed by choosing Delete. For details, see Filtering Out and Removing Unshelved Changes and Restoring Unshelved Changes .
Import Patches		Choose this option to apply patches created externally or through PhpStorm. In the dialog box that opens, choose the files to import patches from. The imported patches are treated as shelved changes and are shown in the Shelf tab of the Version Control tool window where you can unshelve them at any time.
Rename	Shift+F6	Choose this option to modify the name of the selected shelved changelist.
Delete	Delete	Choose this option to permanently delete the selected shelved change. Tiy can remove any change no matter whether it has been already unshelved or not. For details, see Filtering Out and Removing Unshelved Changes .
Show Already Unshelved		Choose this option to have PhpStorm show all available shelved changes, both already applied and not. By default, unshelved changes are hidden. For details, see Filtering Out and Removing Unshelved Changes .
Delete All Already Unshelved		Choose this option to have PhpStorm permanently remove all shelved changes that have been already applied. For details, see Filtering Out and Removing Unshelved Changes .

Update Info Tab

View | Tool Windows | Version Control

This tab is available when local information is synchronized to the server.

Item	Tooltip and shortcut	Description
	Group by Packages	When this button is pressed, the update information within nodes is grouped by namespaces.
	Group by Changelists	When this button is pressed, the update information within nodes is grouped by changelists, and by the day the changelist has been committed. The Update Info tab is divided into two panes: the left pane shows the changelists, grouped by the check-in date, and the right pane shows the list of changed files . Grouping by changelists is not available if the project is under Git or Mercurial control.
	Expand all Ctrl+NumPad Plus Collapse all Ctrl+NumPad -	Use these buttons to have all nodes expanded or collapsed.
	Show Diff Ctrl+D	Click this button to open the Differences viewer for files , where you can compare the local copies of all the project files one after another with their updates from the server. Use the buttons Compare Next File and Compare Previous File to scroll through the list of updated files.
	Close Ctrl+Shift+F4	Click this button to close the tab.
	Help F1	Click this button to show reference page.
The controls of the Changed files pane appear when the button is pressed.		
	Show Diff Ctrl+D	Click this button to show differences between the selected and the previous revision for the selected file in the Changes Files pane.
	Show Diff with Local	Click this button to show differences between the selected revision of a file, and its current local copy.
	Edit Source F4	Click this button to open local copy of the selected file for editing.
	Open Repository Version	Click this button to open repository version of the selected file.
	Revert Selected Changes	Click this button to revert selected changes.
		This button is only available for the files under Subversion, and enables viewing differences in properties of the selected file.
	Show History	Click this button to open the History view of the selected file in the History Tab of the Version Control tool window.
	Group by Directory Ctrl+P	When the button is not pressed, the pane shows a flat list of files. When the button is pressed, the pane shows files in their respective packages. In the latter case, expand and collapse buttons appear in the toolbar.
	Expand all Ctrl+NumPad Plus Collapse all Ctrl+NumPad -	Use these buttons to have all nodes expanded or collapsed.
	Select All Ctrl+A	Click this button to select all files in the Changed Files pane.

MQ: <project_name> Tab

View | Tool Windows | Version Control - Log - Context menu of a commit - Mercurial - Goto patch

This tab is only available if your project is under Mercurial version control.

It displays all patches and allows you to manage the Mercurial Queue .

You can drag-and-drop patches in the queue to change the order in which they will be applied.

Toolbar and Context Menu

Item	Shortcut	Description	Available from
Reload from file	Ctrl+F5	Use this command to refresh the list of patches from the `series` file. Note that when you drag-and-drop patches in the queue, the corresponding changes in the `series` file will only be saved when you switch to a different tab, or perform some other external action. This allows you to revert such changes simply by reloading from file.	Toolbar
Goto	Shift+Alt+G	Use this command to apply the selected patch and all patches above it in the queue. The applied patches will become visible in the Log tab and the changes will be registered in the working copy of the repository.	Toolbar Context menu
Move and Push	Shift+Alt+P	Use this command to apply the selected patch without applying whatever other patches may be before it in the patch stack.	Toolbar Context menu
Fold	Shift+Alt+D	Use this command to merge the selected non-applied patch with the topmost applied patch, and remove it from the list.	Toolbar Context menu
Delete	Delete	Use this command to delete the selected patch from the series file	Toolbar
Apply Patch	N/A	Use this command to apply the selected patch. The Apply Patch dialog will be displayed where you can select which changes you want to restore.	Context menu

V8 Heap Tool Window

The tool window opens when you take a snapshot and choose to open it. The tool window shows the collected profiling data. If the window is already opened and shows the profiling data for another session, a new tab is added. Tabs that were opened automatically are named after the run configurations that control execution of the applications and collecting the profiling data.

If you want to open and analyze some previously saved mempry profiling data, choose V8 Profiling - Analyze V8 Heap Snapshot on the main menu and select the relevant `.snapshot` file. PhpStorm creates a separate tab with the name of the selected file.

The tool window has three tabs that present the collected information from difference point of views.

On this page:

- Containment
- Biggest Objects
- Summary
- Details Pane
- Toolbar
- Context Menu of an Object

Containment

The tab shows the objects in you application grouped under several top-level entries: DOMWindow objects, Native browser objects, and GC Roots, which are roots the Garbage Collector actually uses. See Containment View for details.

For each object, the tab shows its distance from the GC root, that is the shortest simple path of nodes between the object and the GC root, the shallow size of the object, and the retained size of the object. Besides the absolute values of the object's size, PhpStorm shows the percentage of memory the object occupies.

Biggest Objects

The tab shows the most memory-consuming objects sorted by their retained sizes . In this tab, you can spot memory leaks provoked by accumulating data in some global object.

Summary

The tab shows the objects in your application grouped by their types. The tab shows the number of objects of each type, their size, and the percentage of memory that they occupy. This information may be a clue to the memory state.

Details Pane

Each tab has a Details pane, which shows the path to the currently selected object from GC roots and the list of object's retainers, that is, the objects that keep links to the selected object. Every heap snapshot has many "back" references and loops, so there are always many retainers for each object.

Navigating through a Snapshot

- To help differentiate objects and move from one to another without losing the context, mark objects with text labels. To set a label to an object, select the object of interest and click on the toolbar or choose Mark on the context menu of the selection. Then type the label to mark the object with in the dialog box that opens.

- To navigate to the function or variable that corresponds to an object, select the object of interest and click on the toolbar or choose Edit Source on the context menu of the selection. If the button and the menu option are disabled, this means that PhpStorm has not found a function or a variable that corresponds to the selected object.
  If several functions or variables are found, they are shown in a pop-up suggestion list.

- To jump from an object in the Biggest Objects or Summary tab or Occurrences view to the same object in the Containment tab, select the object in question in the Biggest Objects or Summary tab and click on the toolbar or choose Navigate in Main Tree on the context menu of the selection. This helps you investigate the object from the containment point of view and concentrate on the links between objects.

- To search through a snapshot:
  1. In the Containment tab, click on the toolbar.
  2. In the V8 Heap Search Dialog that opens, specify the search pattern and the scope to search in. The available scopes are:
     - Everywhere: select this check box to search in all the scopes. When this check box is selected, all the other search types are disabled.
     - Link Names: select this check box to search among the object names that V8 creates when calling the C++ runtime, see http://stackoverflow.com/questions/11202824/what-is-in-javascript .
       In the V8 Heap Tool Window , link names are marked with the `%` character (`%<link name>`).
     - Class Names: select this check box to search among functions-constructors.
     - Text Strings: select this check box to perform a textual search in the contents of the objects.
     - Snapshot Object IDs: select this check box to search among the unique identifiers of objects. V8 assigns such a unique identifier in the format to each object when the object is created and preserves it until the object is destroyed. This means that you can find and compare the same objects in several snapshots taken within the same session.
       In the V8 Heap Tool Window , object IDs are marked with the `@` character (`@<object id>`).
     - Marks: select this check box to search among the labels you set to objects manually by clicking on the toolbar of the Containment tab.

     The search results are displayed in the Details pane, in a separate Occurrences of '<search pattern>' view. To have the search results shown grouped by the search scopes you specified, press the Group by Type toggle button on the toolbar.

     When you open the dialog box next time, it will show the settings from the previous search.

Toolbar

The toolbar is common for all tabs and most of the toolbar buttons are available in all tabs.

ItemDescriptionAvailable

	in	
	Click this button to set a label to the selected object. This helps you differentiate objects and move from one to another without losing the context.	Containment tab, Biggest Objects tab, Summary tab, Details pane, Occurrences view
	Click this button to find a function in a snapshot: in the V8 Heap Search Dialog that opens, specify the search pattern and the scope to search in. The available scopes are: – Everywhere: select this check box to search in all the scopes. When this check box is selected, all the other search types are disabled. – Link Names: select this check box to search among the object names that V8 creates when calling the C++ runtime, see http://stackoverflow.com/questions/11202824/what-is-in-javascript . In the V8 Heap Tool Window , link names are marked with the `%` character (`%<link name>`). – Class Names: select this check box to search among functions-constructors. – Text Strings: select this check box to perform a textual search in the contents of the objects. – Snapshot Object IDs: select this check box to search among the unique identifiers of objects. V8 assigns such a unique identifier in the format to each object when the object is created and preserves it until the object is destroyed. This means that you can find and compare the same objects in several snapshots taken within the same session. In the V8 Heap Tool Window , object IDs are marked with the `@` character (`@<object id>`). – Marks: select this check box to search among the labels you set to objects manually by clicking on the toolbar of the Containment tab.	Containment tab
	Click this button to jump from an object in the Biggest Objects or Summary tab or Occurrences view to the same object in the Containment tab. This helps you investigate the object from the containment point of view and concentrate on the links between objects.	Biggest Objects tab, Summary tab, Occurrences view
	Click this button to navigate to the function or variable that corresponds to the selected object.	Containment tab, Occurrences view
	Press this toggle button to have the search results shown grouped by the search scopes you specified.	Occurrences view
	Click this button to open the reference page for the tool window.	All
	Click this button to close the tool window.	All

Context Menu of an Object

Item	Description
Mark	Choose this option to set a label to the selected object. This helps you differentiate objects and move from one to another without losing the context.
Navigate in Main Tree	Choose this option to jump from an object in the Biggest Objects or Summary tab or Occurrences view to the same object in the Containment tab. This helps you investigate the object from the containment point of view and concentrate on the links between objects.
Jump to Source	Choose this option to navigate to the function or variable that corresponds to the selected object.

V8 Heap Search Dialog

The dialog box opens when you click on the toolbar in the V8 Heap Tool Window Use the dialog box to find constructors, object IDs, character strings, etc. in a snapshot. The search results are displayed in the Details pane, in a separate Occurrences of '<search pattern>' view.

Item	Description
Search	In this text box, type the search pattern to look for. Select the Case Sensitive check box if necessary.
Scope	In this area, specify the type of objects to limit the search to. When the Everywhere check box is selected, all the other search types are not available. – Everywhere: select this check box to search in all the scopes. When this check box is selected, all the other search types are disabled. – Link Names: select this check box to search among the object names that V8 creates when calling the C++ runtime, see http://stackoverflow.com/questions/11202824/what-is-in-javascript . In the V8 Heap Tool Window , link names are marked with the `%` character (`%<link name>`). – Class Names: select this check box to search among functions-constructors. – Text Strings: select this check box to perform a textual search in the contents of the objects. – Snapshot Object IDs: select this check box to search among the unique identifiers of objects. V8 assigns such a unique identifier in the format to each object when the object is created and preserves it until the object is destroyed. This means that you can find and compare the same objects in several snapshots taken within the same session. In the V8 Heap Tool Window , object IDs are marked with the `@` character (`@<object id>`). – Marks: select this check box to search among the labels you set to objects manually by clicking on the toolbar of the Containment tab.

V8 Profiling Tool Window

The tool window is opened automatically when you stop the Node.js application you are profiling. If the window is already opened and shows the profiling data for another session, a new tab is added. Tabs that were opened automatically are named after the run configurations that control execution of the applications and collecting the profiling data.

If you want to open and analyze some previously saved profiling data, choose V8 Profiling - Analyze V8 Profiling Log on the main menu and select the relevant V8 log file `isolate-<session number>`. PhpStorm creates a separate tab with the name of the log file.

Based on the collected profiling data, PhpStorm builds three call trees and displays each of them in a separate pane. Having several call trees provides the possibility to analyze the application execution from two different points of view: on the one hand, which calls were time consuming ("heavy"), and on the other hand, "who called whom".

On this page:

- Toolbar
- Context Menu
- Top Calls Pane
- Bottom-up Pane
- Top-down Pane
- Flame Chart
- Selecting a Fragment in the Timeline
- Synchronization in the Flame Chart

Toolbar

Item	Tooltip	Description	Available in
	Jump to source	Click this button to navigate to the function definition.	Top Calls Bottom-up Top-down
	Filter	Click this button to filter out light calls and have PhpStorm display only the calls that indeed cause performance problems. Using the slider, specify the minimum Total% or Parent% value for a call to be displayed and click Done.	Top Calls Bottom-up Top-down
	Expand Heavy Traces	When a tab for a profiling session is opened, by default the nodes with heaviest calls are expanded. While exploring the trees, you may like to fold some nodes or expand other ones. Click this button to restore the original tree presentation.	Top Calls Bottom-up Top-down
	Expand All/ Collapse All	Click these buttons to expand or collapse all the nodes in the current pane.	Top Calls Bottom-up Top-down
	Export to text file/ Export Timeline Chart	Click this button to save the call tree in the current pane to a text file or the current timeline chart to a `.png` file. Then specify the target file in the dialog box that opens.	All
	Help	Click this button to navigate to the Help topic for the tool window.	All
	Close	Click this button to close the V8 Profiling tool window.	All
	Zoom	Click this button to open the selected fragment of the flame chart in a separate tab and have the selected fragment enlarged to fit the tab width so you can examine the fragment with more details.	Flame Chart

Context Menu

The context menu is available only from items in the Top Calls, Bottom-up, and Top-down panes.

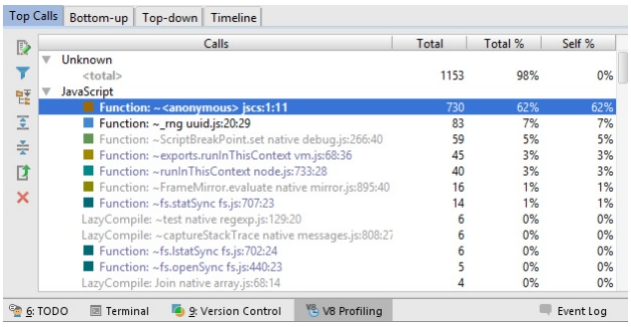
Item	Description
Copy Call	Choose this option to copy the name of the selected function and the name of the file where it is defined to the Clipboard.
Copy	Choose this option to copy the name of the selected function, the name of the file where it is defined, and the measurements data. This may be helpful if you want to compare the measurements for a function from two sessions, for example, after you make some improvements to the code.
Compare with Clipboard	Choose this option to compare the selected with the contents of the Clipboard in the Difference Viewer that opens.

Expand Node/ Choose these options to expand or collapse the selected tree node.

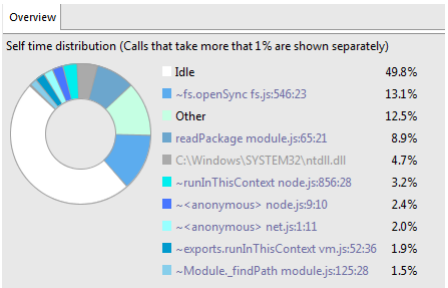
Collapse
Node

Top Calls Pane

The Top Calls pane shows a list of performed activities sorted in the descending order by the Self metrics. For each activity PhpStorm displays its Total, Total%, and Self% metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined.



The diagram in the Overview pane shows distribution of self time for calls with the Self% metrics above 1%.



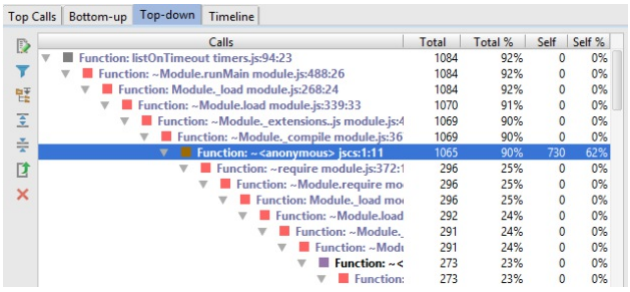
Bottom-up Pane

The Bottom-up pane also shows the performed activities sorted in the descending order by the Self metrics. Unlike the Top Calls pane, the Bottom-up pane shows only the activities with the Total% metrics above 2 and the functions that called them. This is helpful if you encounter a heavy function and want to find out where it was called from. For each activity PhpStorm displays its execution time in ticks and the Of Parent metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined.

Top-down Pane

The Top-down pane shows the entire call hierarchy with the functions that are execution entry points at the top. For each activity PhpStorm displays its Total, Total%, Self, and Self% metrics. For each function call, PhpStorm displays the name of the file, the line, and the column where the function is defined. Some of the functions may have been optimized by V8, see Optimizing for V8 for details.

- The functions that have been optimized are marked with an asterisk (`*`) before the function name.
- The functions that possibly require optimization but still have not been optimized are marked with a tilde (`~`) character before the function name. Though optimization may be delayed by the engine or skipped if the code is short-running, a tilde (`~`) points at a place where the code can be rewritten to achieve better performance.



Flame Chart

Use the multicolor chart in the Flame Chart tab to find where the application paused and explore the calls that provoked these pauses. The chart consists of four areas:

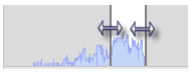
- The upper area shows a timeline with two sliders to limit the beginning and the end of a fragment to investigate.
- The bottom area shows a stack of calls in the form of a multicolor chart. When called for the first time, each function is assigned a random color, whereupon every call of this function within the current session is shown in this color.
- The middle area shows a summary of calls from the Garbage Collector, the engine, the external calls, and the execution itself. The colors reserved for the Garbage Collector, the engine , the external calls, and the execution are listed on top of the area:

- The right-hand pane lists the calls within a selected fragment, for each call the list shows its duration, the name of the called function, and file where the function is defined.

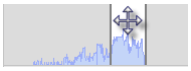
Selecting a Fragment in the Timeline

To explore the processes within a certain period of time, you need to select the fragment in question. You can do it in two ways:

- Use the sliders:



- Click the window between two sliders and drag it to the required fragment:



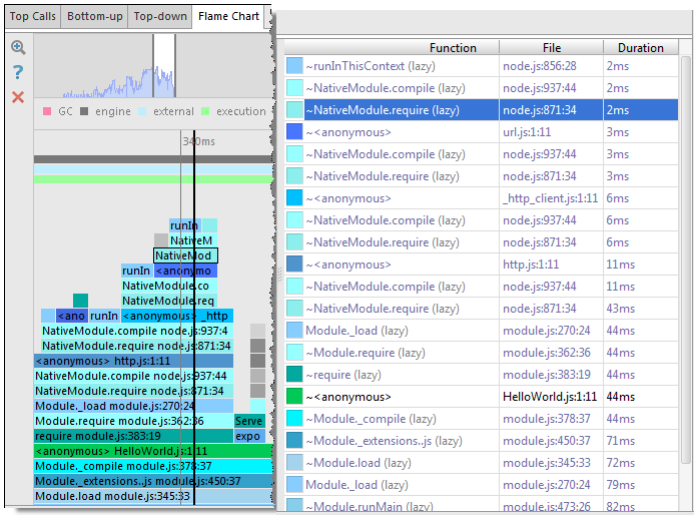
In either case, the multicolor chart below shows the stack of calls within the selected fragment.

To enlarge the chart, click the selected fragment and then click the Zoom button on the toolbar. PhpStorm opens a new tab and shows the selected fragment enlarged to fit the tab width so you can examine the fragment with more details.

Synchronization in the Flame Chart

The bottom and the right-hand areas are synchronized: as you drag the slider in the bottom area through the timeline the focus in the right-hand pane moves to the call that was performed at each moment.

Moreover, if you click a call in the bottom area, the slider moves to it automatically and the focus in the right-hand pane switches to the corresponding function, if necessary the list scrolls automatically. And vice versa, if you click an item in the list, PhpStorm selects the corresponding call in the bottom area and drags the slider to it automatically:



PhpStorm supports navigation from the right-hand area to the source code of called functions, to the other panes of the tool window, and to areas in the flame chart with specific metrics.

- To jump to the source code of a called function, select the call in question and choose Jump to Source on the context menu of the selection.
- To switch to another pane, select the call in question, choose Navigate To on the context menu of the selection, and then choose the destination:
  - Navigate in Top Calls
  - Navigate in Bottom-up
  - Navigate in Top-down

  PhpStorm switches to the selected pane and moves the focus to the call in question.
- To have the flame chart zoomed at the fragments with specific metrics of a call, select the call in question, choose Navigate To on the context menu of the slection, and then choose the metrics:
  - Navigate to Longest Time
  - Navigate to Typical Time
  - Navigate to Longest Self Time
  - Navigate to Typical Self Time

You can also navigate to the stacktrace of a call to view and anlyze exceptions. To do that, select the call in question and choose Show As Stacktrace. PhpStorm opens the stacktrace in a separate tab, to return to the Flame Chart pane, click V8 CPU Profiling tool window button in the bottom tool window.

Version Control Reference

This part provides miscellaneous information, related to common version control operations, and to VCS integrations:

- CVS Reference
- Git Reference
- Mercurial Reference
- Subversion Reference
- Perforce Reference
- Apply Patch Dialog
- Enable Version Control Integration Dialog
- Checkout from TFS Wizard
- Commit Changes Dialog
- Configure Ignored Files Dialog
- Create Patch Dialog
- File Status Highlights
- New Changelist Dialog
- Patch File Settings Dialog
- Push Dialog (Mercurial, Git)
- Revert Changes Dialog
- Select Target Changelist Dialog
- Shelve Changes Dialog
- Show History for File / Selection Dialog
- Show History for Folder Dialog
- Unshelve Changes Dialog

CVS Reference

In this part:

- CVS Global Settings Dialog
- CVS Options Dialog
- CVS Root Dialog
- CVS Tool Window

CVS Global Settings Dialog

File | Settings | Version Control | VCSs | CVS for Windows and Linux

PhpStorm | Preferences | Version Control | VCSs | CVS for OS X

`Ctrl+Alt+S`

Use this dialog box to set up CVS options at the global level. The dialog is available for files and directories that are under CVS control.

Item	Description
Charset	From this drop-down list, select the character set to be used.
Use gzip compression	Select this check box to apply `gzip` compression.
Password file	In this text box, specify the fully qualified path to the `.cvspass` file. Click ... to select the file in the corresponding dialog .
Connection timeout	In this text box, type the timeout value in seconds.
Send environment variable to server	Select this check box to have CVS-related environment variables sent to the server.
Log CVS client/server output to `cvs.log` file	Select this check box to enable logging and have the `cvs.log` log file stored in the `log` directory of your PhpStorm installation.

CVS Roots Dialog

VCS | CVS | Configure CVS Roots

Use this dialog box to set up CVS roots. The dialog box is available for files and directories that are under CVS version control.

Common Options

Item	Description
+	Click this button to configure a new CVS root.
−	Click this button to remove the selected CVS root configuration from the list.
	Click this button to create a copy of the selected CVS root.
Global Settings	Click this button to open the Global CVS Settings dialog box where you can set up CVS options at the global level.
CVS root	In this text box, specify the CVS repository string according to the following syntax: `[:method:][[user][:password]@]hostname[:[port]]/path/to/repository` . Tip Obtain the valid string from your system administrator or click the Edit by Field button to open the Configure CVS Root Field by Field dialog box where you can specify the mandatory connection parameters and have PhpStorm assemble them into a correct repository string.
Edit by Field	Click this button to open the Configure CVS Root Field by Field dialog box where you can specify the mandatory connection parameters and have them assembled into a CVS root string automatically.
Use version	Use this section to specify the revision you want to synchronize your local working copy with. The available options are: − HEAD revision: this option is suggested by default. − By tag: select this option to access the revision with a specific tag. Type the desired tag in the text box or click the Browse button ... and select the desired tag from the list. The list shows all the tags available on the CVS server according to the specified CVS root. − By date: select this option to access the revision with a specific date and time stamp. Type the end date and time in the format `dd:mm:yy hh:mm:ss` or click the Browse button ... and select the desired date from the calendar. This date and time are passed to the server with the GMT parameter. Tip The controls in the area are available only after the CVS root text box is filled in with valid data. Warning! If you perform update or checkout from the CVS repository with the By tag or By date option selected, the resulting working copy will be permanently restricted to the specified tag or date, until you force the update operation to reset this sticky data .
Test connection	Click this button to check that the specified settings ensure successful connection to the CVS server.

Additional Connection Settings

In this area, specify additional settings to flexibly configure connection to the CVS server. The contents of the area depends on the connection method set in the CVS root text box.

− pserver

− ext

− ssh

− local

Pserver

Warning! The settings specified in this area affect all CVS roots that use the pserver connection method.

Item	Description
Password	In this text box, type the fully qualified path to the `.cvspass` file. Click the Browse button ... to select the file in the corresponding dialog .
Connection timeout	In this text box, type the connection timeout in seconds.
Proxy Settings	See the Proxy Settings section below.

Ext

Item	Description
Use internal `ssh` implementation	Select this check box to access the ssh area , where you can specify the SSH version to use, the port to listen to, and configure your private key and password. Clear this check box to access the Ext Protocol Settings area with the following controls available: − Path to external rsh: in this text box, specify the location of the external `rsh` . If necessary, click the Browse button ... to select the necessary location in the corresponding dialog . − Path to private key file: in this text box, specify the location of the file with your private `ssh` key. If necessary, click the Browse button ... to select the file in the corresponding dialog . − Additional parameters: in this text box, specify additional connection parameters.

Ssh

This area is also available when you have specified the ext connection method and selected the Use internal ssh implementation check box.

Item	Description
SSH version	In this area, specify the SSH version to use. The available options are: − Allow both − Force SSH1

– Force SSH2

Port	In this text box, specify the `ssh` port to listen to.
Use Private key file	Select this check box, if you want to pass server authentication using a private `ssh` key. In the text box, specify the location of the file with your private `ssh` key. If necessary, click the Browse button ... to select the file in the corresponding dialog .
Change password	Click this button to open the SSH PAssword dialog box, where you can specify the password for the curent CVS root.
Proxy Settings	See Proxy Settings section below.

Local

Tip PhpStorm does not provide the server functionality. If you want to use a local CVS client, you need to install CVS on your local host computer and configure it to work as a server.

Item	Description
Path to CVS client	In this text box, type the path to CVS client installed on the host computer and configured to work as a server. If necessary, click the Browse button ... to select the necessary location in the corresponding dialog .
Server command	In this text box, specify the server command.

Proxy Settings

Item	Description
Use proxy	Select this check box to enable using the Proxy server and access the Login, Password, Proxy host, and Proxy port text boxes below. This check box is available only in two cases: – The connection method is `pserver` or `ssh` . – The connection method is `ext` and the Use internal ssh implementation check box is selected.
Protocol	Select the protocol to use. The available options are: – HTTP – Socks4 – Socks5
Login	In this text box, specify your user name.
Password	In this text box, specify the user password.
Proxy host	In this text box, specify the Proxy host name.
Proxy port	In this text box, specify the Proxy port number.

CVS Tool Window

VCS | Browse CVS Repository

View | Tool Windows | CVS

This tool window opens, when you browse a CVS repository, and enables you to view contents of the repository, check out files, browse changes, view annotations and navigate to source code.

Item	Description
	Click this button to close the current tab.
	Click this button to open the selected file in the editor.
	Click this button to obtain a local copy of the selected file or directory.
	Click this button to open selected file in the editor with the annotations turned on. See Viewing Annotations .
	Click this button to see the changes that affect the selected file or directory, and that have been committed to the repository by a certain user, or during the specified period. The filtering information is entered in the Search Criteria dialog. Search results show in a dedicated tab of the Version Control tool window .
	Click this button to show reference page.

Import into CVS

VCS | Import into CVS

Use this dialog to import a directory into the specified CVS repository.

Select CVS Configuration

Use this page to select the target CVS root and change its configuration , if necessary.

Select Directory to Import to

Use this page to select the target directory.

Select Import Directory

Use this page to select the directory to be imported. If you are importing an PhpStorm project, make sure that the project file is located under that directory. Multiple selection is not available.

Customize Keyboard Substitution

Use this page to specify the keyword substitution rule for the files imported into the repository.

Import Settings

Item	Description
Name in repository	In this field, specify the name that corresponds to the `module` argument. Tip For import, `module` refers to the absolute location in the repository, not to a module name defined in the modules file.
Vendor	In this field, specify the name that corresponds to the `vendor-tag` argument. This tag is used as a branch tag. No checkouts will ever be done explicitly on it. Type a name that is relevant to the project, or just VENDOR.
Release tag	In this text box, specify the string that corresponds to the `release-tag` argument. The tag should refer to a version or a release number. Tip A tag name cannot contain punctuation marks. For example: `-release-2.2` is wrong `release-2-2` is correct.
Log message	In this field. specify the string that corresponds to the `-m` command-line argument. By default, the field shows the previous log message; you can accept default, or type a new comment.
Checkout after import	Select this option to have CVS checkout run after completing the import operation.
Make checked out files read-only	Select this option to mark the checked out files as read-only after import. This option is disabled if the Checkout after import option is cleared.

Check Out From CVS Dialog

VCS | Checkout From Version Control | CVS

The dialog consists of the following pages:

- Select CS Configuration
- Select CVS Element to Check Out
- Select Checkout Location
- Check out to

Select CVS Configuration

Item	Description
List of available CVS configurations	Use this list to select the desired CVS configuration.
Configure	Click this button to define a new CVS configuration, or modify an existing one, in the CVS Roots dialog box.

Select CVS Element to Check Out

Use this page to select elements of the repository to check out. Next button is only available when an element is selected.

Select Checkout Location

Use this page to specify the target location for the artifacts to check out. All actions can be performed using the toolbar buttons, or context menu.

Item	Shortcut	Description
	Ctrl+1	Navigate to the user home.
	Ctrl+2	Navigate to the project root directory.
	Alt+Insert	Create a new directory, where the files will be checked out to.
	Delete	Delete selected directory.
	Ctrl+Alt+Y	Synchronize with external changes.

Check out to

Use this page to define CVS-specific checkout options.

Item	Description
List of local paths	Select the local path to which the module name should be added.
Make new files read-only	Check this option to set read-only attribute for the files that did not exist locally but were checked out from the repository.
Prune empty directories	Check this option to delete empty directories from the repository.
Change keyword substitution to	Check this option to enable keyword substitution, and select the desired substitution mode from the drop-down list.

Configure CVS Root Field by Field Dialog

VCS | CVS | Configure CVS Roots

This dialog box opens when you click the Edit by Field button in the CVS Roots dialog box. Use this dialog box to specify the parameters for connecting to the CVS server and have PhpStorm assemble them into a correct repository string according to the CVS root string syntax.

Item	Description
Method	From this drop-down list, select the desired connection method. The available options are: – pserver – ext – ssh (internal implementation) – local
User	In this text box, type your login to the CVS server.
Port	In this text box, specify the port to listen to on the CVS server host.
Host	In this text box, type the name of the host where the desired CVS server is located.
Repository	In this text box, type the path to the CVS repository relative to the host name.

Rollback Actions With Regards to File Status

In this section:

- Menu commands according to file status .
- Effect of rolling back local changes .

Menu Commands According to File Status

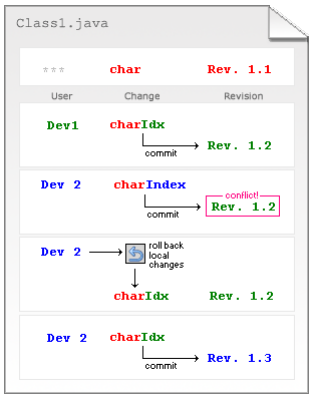
Depending on the file status, the Rollback Changes command will be *aliased* as shown in the following table:

File status	Rollback Command	Result
`modified`	Rollback Local Changes	all changes made in the file will be reverted, and the file will acquire the *up to date* status
`deleted`	Rollback Deletion	file will be restored both on the disk and in CVS with the *up to date* status
`externally deleted`	Rollback Deletion	file will be restored on the disk and assigned the *up to date* status
`added`	Rollback Creation	file will be deleted from disk
`merged`	Rollback Local Changes	all local changes will be dropped, changes from the repository will be accepted, and the file will be assigned the *up to date* status
`merged with conflicts`	Rollback Local Changes	all local changes will be dropped, changes from the repository will be accepted, and the file will be assigned the *up to date* status
`unknown`	Rollback Creation	file will be deleted from the disk

Effect of rolling back local changes

The effect of Rollback Local Changes may not be what you intuitively expect in terms of the revision you have locally after running the command.

The image below represents a file in CVS and a sequence of actions by two developers. It shows a simple example of what happens in terms of the local copy's CVS revision after rolling back conflicting local changes.



Here's what happens:

- The developer *Dev1* takes Revision 1.1 from the repository, modifies it, and commits changes to CVS.
- The developer *Dev2* doesn't know about *Dev1*'s changes and modifies the same code in the local copy of Revision 1.1. When *Dev2* commits these changes, he gets a message from CVS that the repository has changed. So he runs Update to synchronize, and his local copy is then updated to Revision 1.2, and CVS sets the *Merged with Conflicts* status on the file.
- *Dev2* decides to roll back local changes. He is left with a local copy of Revision 1.2 .
- When *Dev2* commits the file to CVS, it becomes Revision 1.3 even though the content is identical to Revision 1.2.

Update Directory / Update File Dialog (CVS)

VCS | CVS | Update File

VCS | CVS | Update Directory Editor | context menu of a file | CVS | Update File

Project Tool Window | context menu of a file | CVS | Update File

Project Tool Window | context menu of a selection | CVS | Update Files

Project Tool Window | context menu of a folder | CVS | Update Directory

Project Tool Window | context menu of a selection | CVS | Update Directories

---

In these dialog boxes, configure synchronization of local files or folders with the repository.

Item	Description
Branch Merging	In this area, specify the repository branch to synchronize with. – Don't merge - select this option to synchronize with the counterpart of the current branch in the repository. This option is selected by default. – Merge with branch - select this option to have the local copy synchronized with a repository branch different from the counterpart of the current branch. In the first text box below, specify the branch to synchronize with. Type the branch name manually or click the Browse button ... and select the desired branch from the list in the Select Tag dialog bos that opens. This option is equivalent to the `-j` command-line option of the `update` command. – Merge two branches - select this option to have the local copy synchronized with the result of merging two repository branches different from the counterpart of the current branch. In the text boxes below, specify the branches to synchronize with. This option is equivalent to the `-j -j` command-line option of the `update` command.

Use Version	In this area, specify the version of repository file(s) or folder(s) to synchronize with.
	– Default - select this option to have the local copy synchronized with the latest repository version.
	– By tag - select this option to have the local copy synchronized with a particular repository version. In the text box, specify the desired version number or tag. Type the version number or tag manually or click the Browse button ... and select the desired branch from the list in the Select Revision or Tag dialog bos that opens.
	  – When updating a single file, you can specify its repository counterpart either through a version number or a tag.
	  – When updating an entire folder, its repository counterpart can be specified only through a tag.
	
	  This option is equivalent to the `-r` command-line option of the `update` command.
	– By date - select this option to have the local copy synchronized with a repository version that was submitted on a particular date. Type the date manually or click the Calendar button and select the desired date from the calendar pop-up window that opens with the current date selected by default.
	This option is equivalent to the `-D` command-line option of the `update` command.

Reset sticky data	Select this check box to remove the date or tag restriction from the local file(s) or folder(s) to be updated. Such restrictions are set on files and folders checked out or previously updated with the By tag or By date options. This option is equivalent to the `-A` command-line option of the `update` command.
Prune empty directories	Select this option while updating a directory to have PhpStorm remove the subfolders whose repository counterparts are empty. This option is equivalent to the `-P` command-line option of the `update` command.
Change keyword substitution to	Select this check box to have the default keyword expansion mode changed. From the drop-down list, choose the relevant substitution. This option is equivalent to the `-k` command-line option of the `update` command.
Create new directories	Select this option while updating a directory to have PhpStorm create new local subfolders when any new subfolders have been created in the repository counterpart of the directory to be updated. This option is equivalent to the `-k` command-line option of the `update` command. By default, all the new subfolders will be picked including empty ones. To avoid creating empty subfolders, it is recommended that you select the Prune empty directories check box as well.
Clean copy	Select this check box to have PhpStorm backup the local changes and replace the changed file(s) with their counterparts from the repository. This option is equivalent to the `-C` command-line option of the `update` command.
Do not show this dialog in the future	Select this option to have the update operation performed silently in the future. To have PhpStorm show this dialog box before update again: 1. Open the Version Control - Confirmation page of the Settings dialog box. 2. In the Display Option dialogs when these commands are invoked area, select the Update check box.

Git Reference

In this part:

- Checkout Dialog
- Clone Repository Dialog
- Line Separators Warning Dialog
- Merge Branches Dialog
- Pull Changes Dialog
- Push Rejected Dialog (Git)
- Rebase Branches Dialog
- Reset Head Dialog
- Git Reset Dialog
- Stash Dialog
- Tag Dialog
- Unstash Changes Dialog
- Update Project Dialog (Git)
- GitHub Integration Reference

Checkout Dialog

VCS | Git | Checkout Branch

Use this dialog box to switch between existing branches and create new branches in a local repository.

Item	Description
Git Root	From this drop-down list, select the path to the local repository in which you want to switch between existing branches or create a new branch.
Current Branch	This read-only field shows the name of the working branch in the selected local repository.
Checkout	From this drop-down list, select the branch or tag to which you want to switch or which you want to copy to a new branch. To specify a particular commit, type its commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`. Refer to the Git commit naming conventions for details.
Validate	Click this button to check that the commit specified in the Checkout field exists and view which files were affected in it. The Validate button next to an editable field indicates that regular expressions are allowed in the field.
Include Tags	Select this check box to have tags also shown in the Checkout drop-down list.
As New Branch	In this text box, type the name of the branch to be created. You can also type the name of an existing branch and select the Override check box.
Create Ref Log	Select this check box to have a reference log for the new branch created where all changes made to the branch references will be recorded.
Track Branch	Select this check box to track the parent of the new branch.
Override	Select this check box if you want to checkout a branch to an existing branch. Use this option with care because the history of the target branch will be lost.
Checkout	Click this button to switch to the specified existing branch or to create a new branch in the local repository.

Clone Repository Dialog

Checkout from Version Control | Git

Use the dialog box to set up a local repository by downloading the data from a remote repository.

Item	Description
Git Repository URL	In this text box, type the URL of the remote repository which you want to clone.
Test	Click this button to check that connection to the remote repository has been established successfully.
Parent Directory	In this text box, specify the directory where you want PhpStorm to create a folder for your local Git repository. Type the path manually or click the Browse button and choose the desired directory in the dialog that opens .
Directory Name	In this text box, type the name of the new folder into which the repository will be cloned. Warning! The parent directory must not contain a folder with the specified name.
Clone	Click this button to start cloning the specified repository.

Line Separators Warning Dialog

The dialog box opens when you attempt to commit a file with `CRLF` line endings, provided that you have enabled this behaviour by selecting the Warn if CRLF line separators are about to be committed checkbox in the Git page of the Settings dialog box.

Item	Description
Cancel	Click this button to stop the commit operation whereupon you can fix the problem manually.
Commit As Is	Click this button to have the file with `CRLF` line separators committed.
Fix and Commit	Click this button to have PhpStorm set the `core.autocrlf` attribute to `input`. As a result, all the `CRLF` separators will be replaced with `LF` separators and committed into the repository. Note that the reverse operation will not be performed when you download the files into your working directory, that is, no `CRLF` will appear in place of `LF`.

Merge Branches Dialog

VCS | Git | Merge Changes

Use this dialog box to specify arguments for merging branches in a local Git repository.

Item	Description
Git Root	From this drop-down list, select the path to the local repository in which you want to merge branches.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository. This is the target branch, the changes from the selected source branches will be applied to it. The contents of the field depend on the selection in the Git Root drop-down list.
Branches to Merge	Use this list box to specify the source branches from which changes will be applied to the target branch. The list shows only those branches that contain applicable commits. Applicable commits are commits made after a branch separated from the target branch.
Strategy	From this drop-down list, select the merge strategy . The available options are: – Default – Resolve - select this option if you need to resolve two HEADs, one of which is the current branch and the other HEAD is the branch which you selected in the Branches to Merge list. When this option is selected, the 3-way merge algorithm is applied. – Recursive - the default merge strategy for merging the current branch with one branch. Select this option if you need to resolve two HEADs by applying the 3-way merge algorithm and there are more than one common ancestor that can be used for 3-way merge. – Octopus - the default merge strategy for merging the current branch with more than one branch. Merges that require resolving conflicts manually are not performed. – Ours - select this option if you need to resolve several HEADs. The result of the merge is always the HEAD of the current branch. – Subtree - a modified recursive strategy. When two or more source branches are selected in the Branches to Merge list box, only the Octopus and Ours options are available.
No Commit	Select this check box if you need to inspect and, if necessary, adjust the result of merging before committing the result. The merge is performed but is not committed automatically, as if it failed.
No Fast Forward	Select this check box to generate a merge commit even if the merge resolved as a fast-forward .
Squash Commit	Select this check box to create a single commit on top of the current branch instead of merging one or more other branches. The working tree and index state are produced as if a real merge happened, but commit is not performed and the HEAD is not moved.
Add Log Information	Select this check box to have PhpStorm populate, in addition to branch names, a log message with one-line descriptions from the actual commits that are being merged.
Commit Message	In this text box, provide a description for the commit. The text box is available only if the No Commit check box is not selected.
Merge	Click this button to initiate merging the specified branches in the local repository according to the defined settings.

Pull Changes Dialog

VCS | Git | Pull Changes

Use this dialog box to specify parameters for fetching changes from a remote repository and applying them to a local repository.

Item	Description
Git Root	From this drop-down list, select the path to the local repository which you want to refresh.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository. The changes retrieved from the source remote repository will be applied to the displayed branch. The contents of the field depend on the selection in the Git Root drop-down list.
Remote	From this drop-down list, select the alias of the source remote repository. Tip To check that the selected alias corresponds to the correct URL address, expand the list - the URL addresses will be displayed explicitly.
Get Branches	Click this button to have the Branches to Merge list box display the actual list of branches in the source remote repository.
Branches to Merge	Use this list box to specify the branches to which you want to apply the fetched changes.
Strategy	From this drop-down list, select the merge strategy . The available options are: – Default – Resolve - select this option if you need to resolve two HEADs, one of which is the current branch and the other HEAD is the branch from which you pulled changes. When this option is selected, the 3-way merge algorithm is applied. – Recursive - the default merge strategy for pulling one branch. Select this option if you need to resolve two HEADs by applying the 3-way merge algorithm and there are more than one common ancestor that can be used for 3-way merge. – Octopus - the default merge strategy for pulling more than one branch. Merges that require resolving conflicts manually are not performed. – Ours - select this option if you need to supersede old development history of side branches. By applying this strategy any number of HEADs can be resolved but the result of the merge is always the HEAD of the current branch. – Subtree - a modified recursive strategy.
No Commit	Select this check box if you need to inspect and, if necessary, adjust the result of merging the fetched data before committing the result. The merge is performed but is not committed automatically, as if it failed.
No Fast Forward	Select this check box to generate a merge commit even if the merge resolved as a fast-forward .
Squash Commit	Select this check box to create a single commit on top of the current branch instead of merging one or more other branches. The working tree and index state are produced as if a real merge happened, but commit is not performed and the HEAD is not moved.
Add Log Information	Select this check box to have PhpStorm populate, in addition to branch names, a log message with one-line descriptions from the actual commits that are being merged.
Pull	Click this button to initiate fetching changes from the specified remote repository and applying them locally according to the defined settings.

Push Rejected Dialog (Git)

VCS | Git | Push

The dialog box opens when pushing a branch is rejected due to lack of synchronization between your local repository and the remote storage. Use this dialog box to have PhpStorm update the local branch using rebase or merge and to configure automatic update of conflicting branches in the future.

Item	Description
Update not rejected repositories as well	– Select this check box to have all the local repositories updated, no matter whether `push` has been rejected for all of them or not. – When this check box is cleared, PhpStorm will update only the repositories for which `push` was rejected. The check box is available only if your project uses several repositories.
Remember the update method choice and silently update in future	When this check box is selected, PhpStorm saves the update method `rebase` or `merge` invoked by pressing the corresponding button and will apply it to update the conflicting local branch silently. After you leave the dialog box, the Auto-update if push ... rejected check box in the Git page of the Settings dialog box is selected and the applied update method becomes default. To change this setting, clear the Auto-update if push ... rejected check box.
Merge	Click this button to have the conflicting branch updated using `merge`.
Rebase	Click this button to have the conflicting branch updated using `rebase`.

Rebase Branches Dialog

VCS | Git | Rebase

Use this dialog box to specify the branch to rebase, the new base, the rebasing mode, and configure the rebasing procedure.

Item	Description
Git Root	From this drop-down list, select the path to the local repository in which you want to rebase a branch.
Branch	From this drop-down list, select the branch to rebase. By default, the current working branch is selected. If you specify another branch, it will be automatically checked out first.
Interactive	Select this check box to view and possibly edit a list of commits to be rebased.
Preserve Merges	Select this check box to have the possibility to recreate merges instead of ignoring them. The check box is available only when the Interactive check box is selected. **Warning!** When this check box is selected, Git does not support squashing commits.
Onto	Use this drop-down list to specify the new base for the selected branch. To specify the required commit, type its commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`. Refer to the Git commit naming conventions for details. If no commit is specified, the HEAD of the selected branch is used as the new base.
Validate	Click this button to check that the commit specified in the Onto field exists and view which files were affected in it.
From	Use this drop-down list to specify the commit starting from which you want to apply the branch to the new base. Type the required commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`. Refer to the Git commit naming conventions for details. To apply the entire branch, leave the field empty.
Validate	Click this button to check that the commit specified in the From field exists and view which files were affected in it.
Show Tags	Select this check box to have tagged commits included in the Onto and From drop-down lists.
Show Remote Branches	Select this check box to have branches in the remote repository included in the Onto drop-down list.
Merge Strategy	From this drop-down list, select the merge strategy. The available options are: – Default – Resolve - select this option if you need to resolve two HEADs, one of which is the current branch and the other HEAD is the branch from which you pulled changes. When this option is selected, the 3-way merge algorithm is applied. – Recursive - the default merge strategy for pulling one branch. Select this option if you need to resolve two HEADs by applying the 3-way merge algorithm and there are more than one common ancestor that can be used for 3-way merge. – Octopus - the default merge strategy for pulling more than one branch. Merges that require resolving conflicts manually are not performed. – Ours - select this option if you need to supersede old development history of side branches. By applying this strategy any number of HEADs can be resolved but the result of the merge is always the HEAD of the current branch. – Subtree - a modified recursive strategy.
Do not use merge strategies	When this check box is selected, no merge strategy is applied during rebase.
Rebase	Click this button to initiate rebasing according to the defined settings.

Interactive Rebase Dialog

VCS | Git | Rebase

The dialog box opens when you click Rebase in the Rebase branches dialog box, with the Interactive check box selected.

Use this dialog box to define the order in which commits should be applied, squash or edit commits before applying, skip commits that contain extraneous changes, etc.

Item	Description
Action	Use this drop-down list to define the action that must be applied to the selected commit. The available options are: – Pick: select this option to apply the selected commit as is. – Edit: select this option to edit the files and/or the commit message before applying the selected commit. – Skip: select this option to ignore the selected commit. – Squash: select this option to combine the selected commit with the previous commit. – Reword: select this option if you want to edit the commit message for the selected commit before applying it. – Fixup: select this option to combine the selected commit with the previous one, and construct a commit message from the previous commit message with the "fixup!" prefix.
Commit	This read-only field displays the hash of the selected commit.
Comment	This read-only field shows the comment supplied for the selected commit.
View	Click this button to view the files affected in the selected commit.
Move Up/Move Down	Use these buttons to change the order in which commits should be applied.

Reset Head Dialog

VCS | Git | Reset Head

Use this dialog box to specify the commit you want to reset the current HEAD to and define the reset method.

Item	Description
Git Root	In this drop-down list, select the path to the local repository in which you want to reset the HEAD of a branch.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository. The contents of the field depend on the selection in the Git Root drop-down list.
Reset Type	Use this drop-down list to define the reset method to use. The available options are: – Mixed - the default strategy. When this option is selected, the index is reset while the working tree is not, which means that changed files are preserved but not marked for commit. You are presented with a report of what has not been updated. – Soft - when this option is selected, the index and the working tree are not affected, only the HEAD pointer is moved to the specified commit. Your current state with any changes remains different from the commit you are switching to. All the changes are "staged" for committing. – Hard - when this option is selected, both the working directory and the index are changed to the specified commit.
To Commit	In this text box, specify the commit you want to reset the current HEAD to. Type the desired commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the current HEAD and the required commit>` . Refer to the Git commit naming conventions for details.
Validate	Click this button to check that the commit specified in the To Commit field exists and view which files were affected in it.
Reset	Click this button to initiate resetting the HEAD to the specified commit accotding to the defined method.

Git Reset Dialog

View | Tool Windows | Log - Context menu of a selected commit - Reset Current Branch to Here

Use this dialog box to reset the current branch head to the selected commit, and update the working tree and the index according to the selected mode. The following options are available:

- Soft: The modified files will not be changed, and the differences will be staged for commit.
- Mixed: The modified files will not be changed, and the differences will be unstaged. This mode is used by default and is identical to the `git reset` command.
- Hard: The modified files will be reverted to the selected commit, and all uncommited changes will be lost.
- Keep: The modified files will be reverted to the selected commit, but local changes will be preserved.

Stash Dialog

VCS | Git | Stash Changes

Use this dialog box to specify parameters for saving local modifications in a new stash.

Item	Description
Git Root	From this drop-down list, select the path to the local repository in which you want to stash modifications.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository.
Message	In this text box, provide a description of the new stash. This description will be displayed in the Unstash Changes dialog box to help you select the required stash.
Keep Index	Select this check box to have PhpStorm bring the changes staged in the index to your working tree for examination and testing.
Create Stash	Click this button to have the specified local modification saved in a new stash.

Tag Dialog

VCS | Git | Tag Files

Use this dialog box to assign tags to commits and commit objects.

Item	Description
Git Root	In this drop-down list, select the path to the local repository in which you want to tag a commit or a commit object.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository.
Tag Name	In this text box, type the name of the new tag.
Force	Select this check box to assign an existing tag to another commit or object. This check box is enabled only if you type the name of an existing tag in the Tag Name text box whereupon PhpStorm displays an error message.
Commit	In this text box, specify the commit or object which you want to tag. Type the desired commit hash or use an expression, for example, of the following structure: `<branch>~<number of commits backwards between the latest commit (HEAD) and the required commit>`. Refer to the Git commit naming conventions for details.
Validate	Click this button to check that the commit specified in the Commit field exists and view which files were affected in it.
Message	In the text box, provide a description of the commit to be tagged. Tip Fill in this text box to create an annotated tag.
Create Tag	Click this button to have the specified commit or commit object tagged.

Unstash Changes Dialog

VCS | Git | Unstash Changes

Use this dialog box to specify parameters for applying stashed modifications to your working copy.

Item	Description
Git Root	From this drop-down list, select the path to the local repository in which you want to apply the stashed modifications.
Current Branch	This read-only field shows the name of the branch which is currently checked out in the selected local repository.
Stashes	From this list, select the stash that you want to apply. Each item is supplied with a number, an indication of the branch in which it was created, and the description provided during the stash creation.
View	Click this button to open the Paths affected in commit dialog box where you can learn which files are affected in the selected stash.
Drop	Click this button to remove the selected stash from the Stashes list.
Clear	Click this button to remove all the stashes from the Stashes list.
Pop Stash	Select this check box to have the selected stash removed from the list after it is applied.
Reinstate Index	Select this check box to have the stashed index modifications applied as well.
As New Branch	If you want to create a new branch on the basis of the selected stash, type the name of the new branch in this list box. When the list box is filled in, the Pop Stash and Reinstate Index check boxes are selected and disabled.
Apply Stash	Click this button to have the specified modifications applied to the working copy.
Pop Stash	Click this button to have the specified modifications applied to the working copy and then removed from the list. The button is available only if the Pop Stash check box is selected.

Update Project Dialog (Git)

VCS | Update Project

In this dialog box, choose the strategy to synchronize your local repository with the remote storage and to choose the way to clean your working tree before update.

Item	Description
Update Type	In this area, choose the method to use for synchronization. The strategy will be applied to all Git version control roots. The available options are: – Merge: choose this option to have the merge strategy applied. The result is identical with that of running `git fetch ; git merge` or `git pull --no-rebase` . – Rebase: choose this option to have the rebase strategy applied. The result is identical with that of running `git fetch ; git rebase` or `git pull --rebase` . – Branch Default: choose this option to have the default command for the branch applied. The default command is specified in the `branch.<name>` section of the `.git/config` configuration file.
Clean working tree before update	In this area, specify the method to save your changes while cleaning your working tree before update. The changes will be restored after the update is completed. The available options are: – Using Stash: choose this option to have the changes saved in a Git stash , so you can apply patches with stashed changed even outside PhpStorm, because they are generated by Git itself. – Using Shelve: choose this option to have the changes saved on a shelf . Shelving is a PhpStorm internal operation, patches generated from shelved changes are normally applied (unshelved) inside PhpStorm. Applying shelved changes outside PhpStorm is also possible but requires additional steps.
Do not show this dialog in the future	Select this check box to have PhpStorm update your project silently in the future using the specified Update Type and Clean working copy method. To have PhpStorm show this dialog box before update again: 1. Open the Version Control - Confirmation page of the Settings dialog box. 2. In the Display Option dialogs when these commands are invoked area, select the Update check box.

GitHub Integration Reference

In this part:

- Select Repository to Clone Dialog
- Share Project on GitHub Dialog
- Login to GitHub Dialog
- Create Gist Dialog
- Create Pull Request Dialog

Select Repository to Clone Dialog

VCS | Checkout from Version Control | GitHub

Use this dialog box to choose the source remote repository, the folder to clone the source to, and the name of the project to be created around the cloned data.

Item	Description
Repository	From this drop-down list, select the source repository to clone the data from.
Parent Directory	In this text box, specify the directory where the local repository for cloned sources will be created. Type the path to the directory manually or click the Browse button ... and choose the desired directory in the Select project destination folder dialog box that opens.
Directory Name	In this text box, specify the name of the folder where the repository will be created based on the cloned sources.
Clone	Click this button to start cloning the sources from the specified remote repository.

Share Project on GitHub Dialog

VCS | Import into Version Control | Share Project on GitHub

Use this dialog box to create a repository on the GitHub remote storage and publish your project sources there.

Item	Description
New repository name	In this text box, type the name of the repository to be created. By default, PhpStorm suggests the name of the current project.
Description	In this text box, describe the main functionality implemented in the project.
Private	Select this check box to suppress access to the repository for other users. **Warning!** The check box is disabled for free accounts.
Share	Click this button to have PhpStorm initiate repository creation and afterwards upload the project sources to the new repository.

Login to GitHub Dialog

VCS | Import into Version Control | Share Project on GitHub

Use this dialog box to log on the GitHub remote storage using your account credentials or to create a GitHub account if you do not have it yet.

Item	Description
Host	Specify the URL of your GitHub repository.
Auth Type	Use this drop-down list to select how you want to be authenticated on GitHub. − Password. If this option is selected and you have two-factor authentication enabled in your GitHub account settings, you will be asked to enter an authentication code each time PhpStorm requires you to log in to your GitHub account. − Token (recommended by GitHub for authentication from third-party applications, as it does not require PhpStorm to remember your password).
Save password	This option is only available if Password is selected as authentication method. Select this option if you want PhpStorm to remember your password.
Save token	This option is only available if Token is selected as authentication method. Select this option if you want PhpStorm to remember your GitHub authentication token.
Login	In this text box, type your GitHub logon name. This field is only available if Password is selected as authentication method above.
Password	In this text box, type your GitHub account password. This field is only available if Password is selected as authentication method above.
Token	Specify your personal access token. This field is only available if Token is selected as authentication method above.
Sign up	Click this link to open the Sign up for GitHub page where you can create a GitHub account.

Create Gist Dialog

<Context menu of an open file in the editor> | Create gist

<Context menu of a file in the Project view> | Create gist

<Context menu of a folder in the Project view> | Create gist

<Context menu of a group of files or folders selected in the Project view> | Create gist

<Context menu of a console> | Create gist

Use this dialog box to save source code and console output in gists .

Warning! The menu item and the dialog box are only available when you are logged on GitHub and a file is opened in the editor.

Item	Description
Description	In this text box, provide a brief description of the code snippet to be published.

By default, the new gist will be public, that is, visible for all registered GitHub users and your login will be shown as the gist's owner. You can change this default behaviour by selecting the check boxes below.

Private	Select this check box to make the new gist available for you only and invisible for other GitHub users.
Anonymous	Select this check box to make the new gist visible for all registered GitHub users but without displaying your login.
Open in browser	− Select this check box to have the new gist opened in the default PhpStorm browser . − To have PhpStorm display the URL address of the new gist so you can access it later, clear this check box.

Create Pull Request Dialog

VCS | Git | Create Pull Request

Creating pull requests is a nice way to share your code with the community and to follow the collaborative development. With it you can tell others about the changes you've made and ask for comments, review, or just share the knowledge. With all this going on, a pull request appears in the original repository only after approval. A pull request can be prepared right from PhpStorm without switching to the browser.

Item	Description
Base Fork	In this field, specify the repository to apply the changes to. PhpStorm attempts to retrieve all the relevant repositories and shows them in the drop-down list. Choose the repository from the drop-down list or click Select Other Fork and from the Form Owner drop-down list choose the name of the user who is the owner of the target repository.
Base Branch	In this field, specify the branch to apply the changes to. Click Show Diff to view the list of commits to be included in the pull request. To view the details of a commit, select it and switch to the Log tab which shows a list of files included in the selected commit list. See also Merging, Deleting, and Comparing Branches .
Title	In this field, specify the name of your pull request.
Description	In this field, optionally provide a brief description of the changes to be applied through the request.

Mercurial Reference

In this part:

- Clone Mercurial Repository Dialog
- Create Mercurial Repository Dialog
- Merge Dialog (Mercurial)
- New Bookmark Dialog
- Pull Dialog
- Switch Working Directory Dialog
- Tag Dialog (Mercurial)

Clone Mercurial Repository Dialog

Checkout from Version Control | Mercurial

Use the dialog box to set up a local repository by downloading the data from a remote repository.

Item	Description
Mercurial Repository URL	In this text box, type the URL of the remote repository which you want to clone.
Test	Click this button to check that connection to the remote repository has been established successfully.
Parent Directory	In this text box, specify the directory where you want PhpStorm to create a folder for your local Mercurial repository. Type the path manually or click the Browse button ... and choose the desired directory in the dialog that opens .
Directory Name	In this text box, type the name of the new folder into which the repository will be cloned. **Warning!** The parent directory must not contain a folder with the specified name.
Clone	Click this button to start cloning the specified repository.

Create Mercurial Repository Dialog

VCS | Create Mercurial Repository

Use this dialog box to create a local Mercurial repository in the folder of your choice.

Item	Description
Create repository for the whole project	Select this option to have a repository initialized in the project root directory. This option is helpful if you want to put the entire project under Mercurial control.
Select where to create repository	Select this option to have a repository initialized in one of the folders below the project root. With this option, you can have folders of your project under control of different version control systems. In the text box below, specify the folder to create the repository in. Type the path manually or click the Browse button and choose the desired folder in the dialog that opens .

Merge Dialog (Mercurial)

VCS | Mercurial | Merge

context menu of the Editor - Mercurial | Merge

VCS | Mercurial | Branches - <branch name> - Merge

context menu of the Editor - Mercurial | Branches - <branch name> - Merge

Use this dialog box to merge the current working directory to a named branch , light-weight branch (bookmark) , or a specific changeset identified by a tag, hash, or revision number.

By default, Mercurial requires that before merge the current working directory should be clean, that is, it should not contain any uncommitted changes. Otherwise the merge operation fails and PhpStorm shows the corresponding error message. The message also recommends that you clean the current working directory by running the `hg merge <target branch, bookmark, or changeset> -C` to discard the uncommitted changes.

Item	Description
Repository	From this drop-down list, choose the repository to run the merge in. The contents of the Branch, Tag, and Bookmark drop-down lists are updated to show the branches, tags, and bookmarks that are available in the selected repository.
Merge with	In this area, choose the branch, bookmark, or changeset to merge with. – Branch: choose this option to switch to another line of development identified by a branch name and merge to the branch head . Choose the desired branch from the drop-down list which shows all the named branches available in the current repository. – Tag: choose this option to merge to a changeset to which you have previously assigned a tag identifier . Choose the relevant tag from the drop-down list. The list shows both local tags (from `.hg/localtags`) and global tags (from `.hgtags`). – Bookmark: choose this option to switch to another line of development which is identified by a bookmark and merge to its head. Choose the relevant bookmark from the drop-down list which shows all the available light-weight branches in the current repository. – Revision: choose this option to merge to a specific changeset identified by its hash or revision number. In the text box, type the relevant revision number or paste the hash. To copy a hash, open the Log tab of the Version Control tool window, select the relevant branch and revision, and then choose Copy Hash on the context menu of the selection.

New Bookmark Dialog

VCS | Mercurial | Branches | New Bookmark

context menu of the Editor - Mercurial | Branches | New Bookmark

---

Use this dialog box to establish a new light-weight Mercurial branch (bookmark). The bookmark will immediately appear in the Branches pop-up. You can create both active and inactive bookmarks. By default, PhpStorm creates an active bookmark, so you are immediately switched to the new bookmark and it is marked with a tick in the Branches pop-up. If you do not want to want to move your development to the new bookmark right now, you can create an inactive bookmark and switch to it later. Note, that tracking and updating is available only for the bookmark that is currently active. You can have only one active bookmark. For details, see http://mercurial.selenic.com/wiki/Bookmarks and Managing Mercurial Branches and Bookmarks .

Item	Description
Bookmark Name	In this text box, type the name of the book,ark. This identifier will always point at the head of the new light-weight branch as you commit changes. You can use this name to identify the head of the relevant light-weight branch when during update or merge.
Inactive	– Clear this check box to activate the new bookmark and thus enable tracking and updating the light-weight branch the bookmarks identifies. The check box is cleared by default. – Select this check box to have an inactive bookmark created, that is, to remain in the current light-weight branch (bookmark) or named branch and switch to the new bookmark later.

Pull Dialog

VCS | Mercurial | Pull Changesets

Use this dialog box to specify parameters for fetching changes from a remote repository and applying them to a local repository.

Item	Description
Pull From	In this text box, specify the URL address of the remote repository to fetch the changes from.

Switch Working Directory Dialog

VCS | Mercurial | Update to

Use this dialog box to update the current working directory to a named branch , light-weight branch (bookmark) , or a specific changeset identified by a tag, hash, or revision number.

By default, Mercurial requires that before update the current working directory should be clean, that is, it should not contain any uncommitted changes. Otherwise the update operation fails and PhpStorm shows the corresponding error message. The message also recommends that you clean the current working directory by running the `hg update <target branch, bookmark, or changeset> -C` to discard the uncommitted changes.

Item	Description
Repository	From this drop-down list, choose the repository to run the update in. The contents of the Branch, Tag, and Bookmark drop-down lists are updated to show the branches, tags, and bookmarks that are available in the selected repository.
Switch to	In this area, choose the branch, bookmark, or changeset to switch to. – Branch: choose this option to switch to another line of development identified by a branch name and update to the branch head . Choose the desired branch from the drop-down list which shows all the named branches available in the current repository. – Tag: choose this option to update to a changeset to which you have previously assigned a tag identifier . Choose the relevant tag from the drop-down list. The list shows both local tags (from `.hg/localtags`) and global tags (from `.hgtags`). – Bookmark: choose this option to switch to another line of development which is identified by a bookmark and update to its head. Choose the relevant bookmark from the drop-down list which shows all the available light-weight branches in the current repository. – Revision: choose this option to update to a specific changeset identified by its hash or revision number. In the text box, type the relevant revision number or paste the hash. To copy a hash, open the Log tab of the Version Control tool window, select the relevant branch and revision, and then choose Copy Hash on the context menu of the selection.
Overwrite locally modified files (no backup)	If you are going to update to another branch, bookmark, or changeset and you have any uncommitted changes in the current line of development, technically there can be two ways to treat them. The uncommitted changes can be either committed before update or abandoned (cleaned). By default, Mercurial requires that before update the current working directory should be clean, that is, it should not contain any uncommitted changes. Otherwise the update operation fails and PhpStorm shows the corresponding error message. The message also recommends that you clean the current working directory by running the `hg update <target branch, bookmark, or changeset> -C` to discard the uncommitted changes. Use the Overwrite locally modified files (no backup) check box to prevent failures during update when the current working copy is not clean. – Select the check box to abandon any uncommitted local changes. – Clear the check box if you are sure that the current working directory is clean.

Tag Dialog (Mercurial)

VCS | Mercurial | Tag Repository

context menu of the Editor - Mercurial | Tag Repository

Use this dialog box to create a global tag that identifies the tip of a repository, which is the most recently changed heade in this repository. The created tag will be stored in the file `.hgtags` and tracked by Mercurial.

Item	Description
Select repository to tag	From this drop-down list, choose the repository whose tip you want to tag. The list shows the location of the repositories under the project root.
Tag name	In this text box, type the name of the tag. By this name, you will be able to find the tag in the Log Tab of the Version Control tool window.

Subversion Reference

In this part:

- Authentication Required
- Changes Browser
- Check Out from Subversion Dialog
- Configure Subversion Branches
- Create Branch or Tag Dialog (Subversion)
- Import Into Subversion
- Integrate Project Dialog (Subversion)
- Integrate to Branch
- Lock File Dialog (Subversion)
- Mark Resolved Dialog (Subversion)
- Select Branch
- Select Repository Location Dialog (Subversion)
- Set Property Dialog (Subversion)
- Subversion Options Dialog
- Subversion Working Copies Information Tab
- SVN Repositories
- Update Project Dialog (Subversion)

Authentication Required

VCS | Browse Subversion Repository

Use this dialog to specify your credentials and gain access to the Subversion repository. The dialog box is opened, when you add a new repository location, or attempt to browse a repository.

Item	Description
Authentication realm	This read-only area displays the URL and name of the repository.
User name	By default, this field shows the current user name. You can change the name as required.
Password	Type the password, defined for your Subversion account.
Save credentials	Select this check box to preserve the specified user name and password.

Changes Browser

VCS | Integrate Project

The dialog box opens when you select the Specified options and click the ... button.

Use the dialog box to select which revision to use in integration.

Check Out From Subversion Dialog

VCS | Checkout From Version Control | Subversion

Use this dialog box to create local working copies.

In this topic:

- Toolbar
- Main Controls
- Context Menu

Toolbar

Item	Tooltip and Shortcut	Description
+	Add Repository Location	Click this button to configure a new repository location .
	Edit Location URL	Click this button to edit the URL address of the selected repository.
−	Discard Location	Click this button to discard selected repository location.
	Show/Hide Details	Click this button to display the details for each node below the repository location (changelist number, user name, date and time of the last change).
	Refresh Ctrl+F5	Click this button to refresh the view.
	Collapse All Ctrl+NumPad -	Click this button to have all the nodes below all the repository locations collapsed.

Repositories

In this area, manage the available repositories and select the locations to check out contents from.

Item	Description
Repositories	Use this tree view to explore and manage the available repository locations. If necessary, right-click a node and choose the relevant item from its context menu.
Checkout	Click this button to check out the contents of the selected node to the specified location.

Context Menu

Item	Description
New	Select this menu item to configure a new repository location , or a new remote folder in the selected repository location.
Checkout	Select this menu item to check out the contents of the selected node.
Checkout	Select this menu item to check out the contents of the selected node.
Compare With	Select this menu item to compare the selected node with the specified branch.
Browse Changes	Select this menu item to view changes that match the specified criteria (author, time range, and revision).
Export	Select this menu item to export the contents of the selected repository or folder to the specified destination. The exported contents are not under version control.
Branch or Tag	Select this menu item to create a branch or tag of the selected folder.
Move or Rename	Select this menu item to change name of the selected folder.
Delete	Select this menu item to delete the selected folder from the repository location.
Copy URL Ctrl+Alt+Insert	Select this menu item to put the URL string to the clipboard.
Refresh	Select this menu item to synchronize to the repository.
Edit Location URL	Select this menu item to edit the selected repository location.
Discard Location	Select this menu item to discard the selected repository location.

SVN Checkout Options Dialog

VCS | Checkout From Version Control | Subversion

The dialog box is the third page of the Checkout From Subversion Wizard. In this dialog box, specify the revision to check out and configure the check-out procedure.

Item	Description
Checkout	This read-only field shows the selected source repository.
Destination	From this list, select the directory to create the working copy in. Choose one of the available folders or click the Browse button ... and select the relevant folder in the dialog box that opens.
Update / Switch to revision	In this area, specify the revision to check out. The available options are: – HEAD - select this option to have the latest revision checked out. – Specified - select this option to have PhpStorm check out a specific earlier revision. Type the revision number in the text box or click the Browse button ... and select the relevant revision from the Changes Browse that opens.
Depth	Use this drop-down list to specify the range of recursion into subdirectories. The available options are: – Empty - select this option to involve only the current file. – Files - select this option to involve the files in the folder. – Immediates - select this option to involve direct children of the current file. – Infinity - select this option to enable full recursion.
Include external locations	Select this check box to have externals included in the working copy.

Configure Subversion Branches

Select Branch pop-up dialog box - Configure Branches

Use this dialog box to compose a list of branches you work with.

Item	Description
Trunk location	In this text box, specify the URL address of the trunk in the repository. If necessary, click the Browse button ... to open the Select Repository Location dialog box and select the required trunk in the repository structure tree.
Branch locations	In this list, select the URL address of the folders in which the required branches are stored.
Add	Click this button to add a branch to the list. The Select Repository Location dialog box opens where you can select the required branch in the repository structure tree.
Remove	Click this button to remove the selected branch from the list.

Create Branch or Tag Dialog (Subversion)

VCS | Subversion | Create Branch or Tag

In this dialog box, set the arguments for creating a branch or a tag on the basis of a local working copy or a repository version.

Item	Description
Copy from	In this section, specify the source folder to create a branch or tag from. The source of the copy can be taken from the local working copy or from the repository.
Working Copy	Click this option to create a branch or tag on the basis of your local working copy. Type the path in the text box or click the Browse button and select the desired directory in the dialog that opens .
Repository Location	Click this option to create a branch or tag on the basis of the repository. Do one of the following: – Type the URL of the repository location in the text box. – Click the browse button and select the source repository location. – Click the button to use the project home directory.
Revision	In this section, specify the source revision to create a branch or tag from. The available options are: – HEAD - select this option to have a branch or tag created on the basis of the HEAD revision. – Specified - select this option to have a branch or tag created on the basis of a specific revision. Type the revision number in the text box manually or click the Browse button and select the desired revision in the Changes Browser dialog box, that opens.
Copy to	Use this section to define the target folder for a branch or tag. The available options are: – Branch or Tag - select this option to have the selected revision copied to a specific branch or tag. In the Base URL text box, specify the base URL of the branch or tag. In the Name text box, specify the name of the new branch. – Any location - select this option to have to have the selected revision copied to a location of your choice. In the text box below, specify the URL of any valid location. Type the URL manually or click the Browse button and specify the desired location in the Select Repository Location dialog box, that opens.
Comment	Type some meaningful description in the text area.

Import into Subversion

VCS | Import into Subversion

Use this dialog box to specify the options for importing data into Subversion.

In this topic:

- Toolbar
- Main Controls
- Context Menu

Toolbar buttons

Item	Tooltip and Shortcut	Description
+	Add Repository Location	Click this button to configure a new repository location .
	Edit Location URL	Click this button to edit the URL address of the selected repository.
−	Discard Location URL	Click this button to discard the selected repository location and remove it from the list.
	Show/Hide Details	Click this button to display the details for each node below the repository location (changelist number, user name, date and time of the last change).
	Refresh Ctrl+F5	Click this button to refresh the view.
	Collapse All Ctrl+NumPad -	Click this button to have all the nodes below all the repository locations collapsed.

Main Controls

Item	Description
Repositories	Use this tree view to explore and manage the available repository locations. Right-click nodes and examine context menus.
Import	Click this button and in the dialog that opens , select the directory whose contents should be imported to the selected repository location.

Context Menu

Item	Description
New	Select this menu item to configure a new repository location , or a new remote folder in the selected repository location.
Checkout	Select this menu item to check out the contents of the selected node.
Compare With	Select this menu item to compare the selected node with the specified branch.
Browse Changes	Select this menu item to view changes that match the specified criteria (author, time range, and revision).
Export	Select this menu item to export the contents of the selected repository or folder to the specified destination. The exported contents are not under version control.
Branch or Tag	Select this menu item to create a branch or tag of the selected folder.
Move or Rename	Select this menu item to change name of the selected folder.
Delete	Select this menu item to delete the selected folder from the repository location.
Copy URL Ctrl+Alt+Insert	Select this menu item to put the URL string to the clipboard.
Refresh	Select this menu item to synchronize to the repository.
Edit Location URL	Select this menu item to edit the selected repository location.
Discard Location	Select this menu item to discard the selected repository location.

Integrate Project Dialog (Subversion)

VCS | Integrate Project

Use this dialog box to integrate differences between two branches in the Subversion repository into a local working copy.

Item	Description
Source 1	In this text box, specify the URL address of the first branch to compare. If necessary, click the Browse button ... and select the desired URL from the Select Repository Location dialog box.
Source 2	In this text box, specify the URL address of the second branch to compare. If necessary, click the Browse button ... and select the desired URL from the Select Repository Location dialog box.
Revision	For each source, specify the revision to use. The possible options are: – HEAD - select this option to use the Head revision of the source. The Head revision is suggested by default – Specified - select this option to use a revision different from the Head revision. Specify the required revision in the text field. If necessary, click the ... button and select the revision from the Changes Browser dialog box.

Tip Based on the sources and revisions you specify, the difference between source 2 and source 1 is calculated and applied to the local working copy.

Use ancestry	Select this check box to take the ancestry of the Source1 and Source2 URLs into consideration when comparing revisions. If the check box is not selected, only the contents of the files are compared.
Try merge, but make no changes	Select this check box to enable the `--dry-run` switch of the `svn` command. If this check box is not selected, the sources are merged silently.
Depth	Use this drop-down list to specify the range of recursion into subdirectories. The available options are: – Empty - select this option to involve only the current file. – Files - select this option to involve the files in the folder. – Immediates - select this option to involve direct children of the current file. – Infinity - select this option to enable full recursion.

Integrate to Branch

View | Tool Windows | Version Control | Repository Tab | Context menu of a changelist or file |

Subversion | Integrate to Branch

Use this dialog to specify the options for integrating changes into a branch.

ItemTooltip	Description
Source branch URL	This read-only field shows the URL address of the source branch.
Target branch URL	This read-only field shows the URL address of the target branch.
Integrate into working copy	From this list, select the path to the local working copy into which the changes will be integrated.
Ignore whitespaces	Select this option if whitespaces are not important.
Try merge, but make no changes	Select this option to preview merge results by enabling the `--dry-run` switch of the `svn` command. If this option is unchecked, sources are merged silently.
+ Add	Click this button to add a working copy to the list.
− Remove	Click this button to remove the selected working copy from the list.

Lock File Dialog (Subversion)

VCS | Subversion | Lock

Context menu of a file | Subversion | Lock

Use this dialog box to lock file when it is necessary to avoid overwriting changes.

Item	Description
Lock Comment	In this text box, describe the reason for locking the file and some additional comments, if necessary.
Steal existing lock	Select this check box to override the lock previously set on the desired file by someone else.
Do not show this dialog in the future	Select this check box to suppress displaying this dialog box and have files and folders locked silently. To have PhpStorm show this dialog box before locking files or folders again: 1. Open the Version Control - Confirmation page of the Settings dialog box. 2. In the Display Option dialogs when these commands are invoked area, select the Checkout check box.

Mark Resolved Dialog (Subversion)

VCS | Subversion | Mark Resolved

Project tool window | context menu of a file | Subversion | Mark Resolved

Local Changes tab of the Version Control tool window | context menu of a file | Subversion | Mark Resolved

Project tool window | context menu of a file | Subversion | Mark Resolved

Version Control tool window - Merged with conflicts list | context menu of a file | Subversion | Mark Resolved

Use this dialog box to have PhpStorm consider conflicts in a file or directory resolved. This operation is most often required after merging text or property conflicts manually.

Item	Description
Files and directories	The list shows all the files and directories where merge or updated resulted in conflicts that PhpStorm cannot resolve automatically. When you have examined the conflicts resolved them manually or considered irrelevant, you need to tell PhpStorm that these files are no longer conflicting. To appoint a file for marking as free from conflicts, select the check box next to it.
Select All	Click this button to have all the items in the list appointed for marking as resolved.
Deselect All	Click this button to clear the list of candidates for marking as resolved.
Mark Resolved	Click this button to have PhpStorm treat all the selected items as conflict free. The dialog box closes, whereupon the Local Changes tab of the Version Control tool window shows the affected files as updated and available for submitting to the server.

Select Branch

Version Control tool window | Repository Tab | Merge Info Pane - Browse

Version Control tool window | Subversion Working Copies Information Tab | Merge from

Update Project/Directory dialog box - Browse

The pop-up dialog box opens when you click the Browse button ... or press `Shift+Enter` to select the path to the target branch.

Use this dialog box to select the relevant branch or working copy.

Item	Description
Trunk	Choose this option to set the current trunk as the target branch or working copy.
Branches	Choose this option to select the relevant branch in the Branches list.
Tags	Choose this option to select the relevant tag in the Tags list.
Configure Branches	Choose this option to open the Configure Subversion Branches dialog box and compose a list of branches you work with.

Select Repository Location Dialog (Subversion)

VCS | Subversion | Branch or Tag | Copy From | Repository location

VCS | Subversion | Branch or Tag | Copy To | Any location

Item	Description
Repositories	Use this tree view to explore and manage the available repository locations. Right-click nodes and examine context menus.
Copy as	Specify the name under which the file or folder will be stored in branch.

Set Property Dialog (Subversion)

VCS | Subversion | Set Property

Use this dialog to define SVN-specific properties for the files and folders under SVN version control (ignore list, externals etc.)

Item	Description
Property name	Enter custom property name in the text field, or use the drop-down list to select one of the pre-defined properties.
Set property value	Click this radio-button to set value for the specified property name in the text area. Properties that accept multiple values, such as an ignore list, can be entered on multiple lines.
Delete property	Click this radio-button to remove selected property from the list.
Update properties recursively	Check this option, if you want to apply the property to every file and directory under the selected directory.

Subversion Options Dialog

File | Settings | Version Control | Subversion for Windows adn Unix

PhpStorm | Preferences | Version Control | Subversion for OS X

Item	Description
Use system default Subversion configuration directory	Store Subversion configuration files in the system default directory: `user_home\Application Data\Subversion`
Subversion configuration directory	Remove the content of the corresponding directory in the Subversion configuration directory. You may need to clear the authorization information from the configuration file, for example, when your credentials have changed.
Clear authentication cache	Remove the content of the corresponding directory in the Subversion configuration directory. You may need to clear the authorization information from the configuration file, for example, when your credentials have changed.

Subversion Working Copies Information Tab

View | Tool Windows | Version Control - Subversion Working Copies Information

Use this tab to configure the format of your working copies.

Tip The tab is only available, when the current project sources are entirely or partially under Subversion control.

The tab displays a list of all detected directories under Subversion control supplied with information on the formats used.

Item	Description
Refresh	Click this button to get the information on all the detected Subversion working copies up-to-date.
Root Path	This read-only field shows the full path to the directory.
URL	This read-only field shows the URL address of the remote directory the selected local copy is mapped to.
Format	This read-only field shows the actual Subversion format used in the selected directory.
Change	Click this link to open the Convert Working Copy Format dialog box, where you can select the desired format option.
Depth	This read-only field shows the range of recursion into subdirectories specified in the Update dialog box.
Working Copy Root	This read-only field is displayed only if the directory in question is the root of a working copy.
Configure Branches	Click this link to open the Configure Subversion Branches dialog box, where you can view and update the list of branches to work with.
Merge from	Click this link to open the Select Branch pop-up dialog box and appoint the source of changes to merge to the current directory.

SVN Repositories

VCS | Browse Subversion Repository

Use this tool window to view, add, and edit the location of SVN repositories.

Toolbar

Item	Tooltip and shortcut	Description
	Add Repository Location	Click this button to configure a new repository location . The New Repository Location dialog box opens, where you can select a repository URL in the drop-down list that contains previously added URL addresses.
	Edit Location Url	Click this button to edit the URL of the selected repository.
	Discard Location	Click this button to remove the selected repository from the list.
	Show/Hide Details	Click this button to display the details for each node under the repository location (changelist number, user name, date and time of the last change).
	Refresh Ctrl+F5	Click this button to refresh the view.
	Expand All/Collapse All Ctrl+NumPad Plus	Click this button to expand/collapse all nodes.
	F1	Click this button to show the corresponding PhpStorm help page.
	Close	Click this button to close the tool window.

Context Menu

Item	Description
New	Select this menu item to configure a new repository location , or a new remote folder in the selected repository location. The New Repository Location dialog box opens where you can select a repository URL in the drop-down list that contains previously added URL addresses.
Show History	Select this item to open the Version Control tool window with the history of the selected repository location.
Checkout	Select this menu item to check out the contents of the selected node.
Compare with	Select this menu item to compare the selected node with the specified branch.
Browse changes	Select this menu item to view changes that match the specified criteria (author, time range, and revision).
Export	Select this menu item to export the contents of the selected repository or folder to the specified destination. The exported contents are not under version control.
Branch or Tag	Select this menu item to create a branch or tag of the selected folder.
Move or Rename	Select this menu item to change name of the selected folder.
Delete	Select this menu item to delete the selected folder from the repository location.
Copy URL	Select this menu item to put the URL string to the clipboard.
Refresh	Select this menu item to synchronize to the repository.
Edit location Url	Select this menu item to edit the selected repository location.
Discard location	Select this menu item to discard the selected repository location.

Update Project Dialog (Subversion)

VCS | Update Project

`Ctrl+T`

Context menu of a file or directory | Subversion | Update File/Directory

VCS | Subversion | Update File/Directory

Use this dialog box to update the local working copy of a file, directory, or project with a revision from the repository.

Item	Description
Update/Switch to specific Url	– Select this check box to synchronize your local working copy with a specific repository. Specify the source repository either in the URL text box through its full Url address or in the Use Branch text box through the branch name. – Clear this check box to bring the changes from the repository that corresponds to the current working copy.
Use Branch	In this text box, specify the location of the required repository through its branch name. Click the Browse button ... to choose the required branch in the Select Branch dialog box that opens. **Note** 1. To use this method of specifying the required repository, you need to configure a list of branches you work with. If you have not done it yet, click Configure Branches in the Select Branch dialog box. 2. The text box is enabled only when the Update/Switch to specific Url check box is selected.
Url	In this text box, specify the location of the required repository through its full URL address. Click the Browse button ... to open the Select Repository Location dialog box. The text box is enabled only when the Update/Switch to specific Url check box is selected.
Update/Switch to specific revision	Select this check box to synchronize your local working copy with a specific revision different from the HEAD revision. The Update/Switch to specific revision text box becomes enabled. In this text box, specify the number of the revision to be used. Click the Browse button ... to open the Changes Browser dialog box. By default, PhpStorm suggests to update your local working copy the HEAD revision. This option corresponds to the `--non-recursive (-N)` switch of the Subversion `update` command.
Depth	Use this drop-down list to specify the range of recursion into subdirectories. The available options are: – Working copy - select this option to get files/directories from repository subtrees that have not been checked out yet. – Empty: select this option to involve only the current file. – Files: select this option to involve the files in the folder. – Immediates: select this option to involve direct children of the current file. – Infinity: select this option to enable full recursion.
Force Update	Select this check box to have local files replaced with the files from the repository even if the local files have modifications and thus abandon the local modifications.
Update administrative information only in changed subtrees	This option only applies to working copies older than SVN 1.7 managed by SVNKit. During synchronization with the server (update), SVN locks your working copy one subtree after another by creating empty `lock` files in the corresponding administrative `.svn` directories. After that, SVN starts comparing file hashes to detect which local files need to be synchronized. When this option is selected, SVN first checks if any files from a subtree have been modified on the server, and locks this subtree (i.e. creates a `.svn/lock` file) only if such files are detected. This approach improves performance but may cause concurrency issues, for example, with antiviral software.
Ignore Externals	Select this check box if you do not want PhpStorm take into account externals definitions during update.
Do not show this dialog in the future	Select this check box to have PhpStorm perform future updates silently. To have PhpStorm show this dialog box before update again: 1. Open the Version Control - Confirmation page of the Settings dialog box. 2. In the Display Option dialogs when these commands are invoked area, select the Update check box.

Perforce Reference

In this part:

- Perforce Options Dialog
- Link Job to Changelist Dialog
- Edit Jobs Linked to Changelist Dialog
- Integrate File Dialog (Perforce)

Perforce Options Dialog

File | Settings | Version Control | Configure | Perforce for Windows and Linux

PhpStorm | Preferences | Version Control | Configure | Perforce for OS X

Ctrl+Alt+S

In this dialog box, configure connection to the specified Perforce server.

Item	Description
Perforce is online	Select this check box to enable establishing connection to the Perforce server. If there is a connection but the server does not respond (for example, because the server is backing up currently), you can disable such attempts by clearing this check box, or PhpStorm will suggest to do that after a timeout. After that the version control-specific operations will be disabled.
Use P4CONFIG or default connection	Use server information and user credentials from the P4CONFIG environment variable or default Perforce client connection. Otherwise, specify port, user, client, and password.
Charset	Select the charset corresponding to the one set on the server.
Dump Perforce commands to *IDEA_Home\bin\p4.output*	Log Perforce commands in the specified file.
Use login authentication	Toggle authentication.
Try to log in silently	Skip prompt dialog. This option works, when login authentication is required.
Use native Perforce API	Speed up connection to the server using a special library.
Path to P4 executable	Specify the path to the Perforce client executable file.
Test connection	Make sure that connection to the Perforce server is established.
Show branching history	See branches for files in the dialogs showing File History. When you work with several branches, it is recommended to enable this option so that the file branches are correctly displayed.

Link Job to Changelist Dialog

The dialog box opens when you click the button in the Edit Jobs Linked to Changelist dialog box.

Use this dialog box to search for available jobs, view their details, and link jobs to the changelist.

> **Tip** When you need to attach only one job to a changelist, you can use the quick search functionality. This requires that you know the exact name of the job or at least can specify a search pattern for it.

Specify search parameters

Use the controls in this area for specifying various criteria to limit the search output. Follow the Perforce jobs syntax rules . The specified values are joined in the generated command line query via the `AND` operation.

> **Tip** At least one of the fields should be filled in.

Item	Description
Job name pattern	In this text box, type the desired job name search pattern.
Status	Use this drop-down list to specify the status of the job you are looking for. The available options are: – * - when this option is selected, all the jobs that match the remaining search criteria are displayed, regardless of their job statuses. **Tip** If you specify Status as the only criterion and select this option, all the jobs that are currently present on the Perforce server will be retrieved. – Open – Closed – Suspended
User name pattern	In this text box, specify the search pattern or the exact name of the user who created the desired job.
Date before/Data after	Use these text boxes to specify the time period the desired job is created in. The appropriate formats are `yyyy/mm/dd` or `yyyy/mm/dd:hh:mm:ss` .
Description pattern	In this text box, type the desired job description search pattern.
Search	Click this button to start searching for jobs that match the specified criteria.

Search results

Use this area to view the details of found jobs, select the desired job, and attach it to the changelist.

Item	Description
Search results	The list contains the jobs found according to the specified search criteria. When you select a job, the read-only area shows its details.
OK	Click this button to link the selected job to the changelist.

Edit Jobs Linked to Changelist Dialog

VCS | Show Changes View - Local

View | Tool Windows | Changes - Local

VCS | Commit Changes

The dialog box opens in the following cases:

- When you select a changelist and then select Edit Associated Jobs from the context menu.
- When you click the button in the Perforce area of the Commit Changes dialog box.

Use the dialog box to to search for Perforce jobs, link jobs to the selected changelist, and detach currently linked jobs.

Item	Tooltip and Shortcut	Description
−	Unlink selected jobs	Click this button to detach the selected job from the changelist.
	Search	Click this button to open the Link Job to Changelist dialog box, where you can search for available jobs, view their details, and link the desired job to the changelist.
+	Find and link job matching the pattern	Click this button to start quick search for the job that matches the pattern specified in the text box and attach the job to the changelist. In the text box, specify the exact name of the desired job or a search pattern according to the Perforce jobs syntax rules .

Tip If only one job matching the pattern is found, it is attached to the changelist automatically. Otherwise, to select a job among several available jobs, click and find the desired job using the Link Job to Changelist dialog box.

Close	Click this button to save the specified settings and leave the dialog box.

Integrate File Dialog (Perforce)

VCS | Integrate Project

Context menu of a file or directory | Perforce | Integrate File/Directory

Use this dialog box to integrate changelists from one branch spec to another.

Item	Description
Branch Spec	Select the branch spec that will be used for change integration. Consider the following: − If the Reverse option is enabled, changes are integrated from the selected branch to the local copy. − If the Reverse option is disabled, changes are integrated from the local copy to the selected branch.
Integrate changelist	Use this option to invoke the Changes Browser, where you can select the changelist that will be integrated into the current branch/local copy.
Store Changes To Changelist	Specify the changelist where the integrated changes should be stored.
Revert unchanged files before sync (p4 revert -a)	Select this option to revert unchanged files.
Run resolve automatically after the sync (p4 resolve -am)	Select this option to automatically resolve the files that can be resolved without conflicts.

Apply Patch Dialog

VCS | Apply Patch

Use the dialog box to restore changes that were preserved in a patch file in the specified directory.

Item	Tooltip and Shortcut	Description
Patch file name		In this text box, specify the name of the *.patch file to be applied. Type the fully qualified name or click the Browse button ... and locate the desired patch file using the Select Patch File dialog box, that opens.
	Group by Directory Ctrl+P	Use this button to toggle between the flat view and the directory tree view. Select the check boxes next to the changes that you want to be applied.
	Expand All/Collapse All Ctrl+NumPad Plus Ctrl+NumPad -	Use these buttons to expand/collapse all nodes
	Map base directory	In the dialog that opens , select the directory relative to which file names in the patch file will be interpreted. You can map a base directory to a single file, directory, or to a selection.
	Show Differences Ctrl+D	Click this button to open the Differences viewer for files that shows the differences between your local working copy, the repository version, and the patch. Use the buttons Compare Previous File and Compare Next File to have the files in patch compared in a chain. Tip If the patch cannot be applied without conflicts, the lines with conflicts are highlighted with red.
	Strip Directory	Use this button to apply the changes to files located in different directories from the ones specified in the patch. Clicking this button removes one slash in the path to the target file. Click the button as many times as many leading directories you need to strip. The number of removed slashes is indicated in square brackets.
	Restore Directory	Use this button to revert the last strip directory action. Click the button as many times as many previously stripped leading directories you need to restore.
	Reset Directories	Use this button to revert all strip directory actions in the selection.
	Remove Directories	Click this button to have all the leading directories stripped and have the changes applied to the file with the specified name in the base directory.
	Refresh	Click this button to synchronize the tree with the current state of the file system.
Summary		This section displays summary information for the currently selected changelist (the number of modified, new, and deleted files).
Existing Changelist		Select this option to add the patched files to an existing changelist and select the desired changelist from the drop-down list.
New Changelist		Select this option to create a new changelist and add the patched files to it. – Name: in this text box, type the name of the new changelist. By default, it is the name of the current patch. – Comment: in in this text box, type the comment to the new changelist. – Make this changelist active: select this check box to have PhpStorm automatically give the active status to the new change list immediately after the changes are restored in it. When this check box is cleared, the current active changelist remains active. See Changelist for details. – Track Context: select this check box to have PhpStorm preserve the context of the task associated with the new changelist on its deactivation and restore the context when the changelist becomes active. See Managing Tasks and Context for details.

Enable Version Control Integration Dialog

VCS | Enable Version Control Integration

In this dialog box, choose one of the registered Version Control Systems to use in your project and assign it to the Project Root.

The dialog box and the menu item are available only if the project does not use any Version Control System.

Item	Description
Select a version control system to associate with the project root	From this drop-down list, select one of the supported version control systems that you want to associate with the project root.

Checkout from TFS Wizard

VCS | Checkout from Version Control | TFS

Use this wizard to download the files from a TFS server according to the settings from a new or an existing workspace .

The menu item and the wizard are available only when the TFS Integration plugin is installed and enabled. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .

In this part:

- Checkout from TFS Wizard: Checkout Mode
- Checkout from TFS Wizard: Source Server
- Checkout from TFS Wizard: Choose Source and Destination Paths
- Checkout from TFS Wizard: Source Workspace
- Checkout from TFS Wizard: Choose Source Path
- Checkout from TFS Wizard: Summary

Checkout from TFS Wizard: Checkout Mode

VCS | Checkout from Version Control | TFS

On this page, choose the way to map the files and folders on your server with their local copies. These mappings are referred to as workspaces . You can either specify new mappings and have a new workspace generated or use mappings from an existing workspace.

Item	Description
Create workspace automatically	Choose this option to have PhpStorm generate a workspace for you, with the folder mapping based on your choice of the local and remote paths .
Workspace name	In this text box, type the name of the workspace to be generated.
Choose workspace manually	Choose this option to have the data downloaded according to the mappings from an existing workspace.

Checkout from TFS Wizard: Source Server

VCS | Checkout from Version Control | TFS - Create workspace automatically

On this page, choose the server to download the data from.

Item	Description
Team servers	In this list box, manage the list of the servers you have access to and select the server to download the data from.
Add	Click this button to open the Add Team Foundation Server dialog box, and specify the address of your TFS server and the credentials to connect to it.
Remove	Click this button to delete the selected server from the list.
TFS Proxy	Click this button to open the Set TFS proxy for server... dialog box where you can specify the parameters for accessing the selected server via Proxy.

Checkout from TFS Wizard: Choose Source and Destination Paths

VCS | Checkout from Version Control | TFS - Create workspace automatically

The page opens when you select the server to download the data from on the Source Server page and click Next.

On this page, specify the remote folder to download the data from and map it to a local folder where the data will be downloaded to. The data is downloaded recursively, that is, the structure of subfolders under the selected source node is reproduced locally.

Item	Description
Source Path	In this area, specify the folder on the server to download the data from. Select the folder in the tree, and Source Path read-only field displays the path to it relative to the server root.
Destination path	In this text box, specify the local folder to download the sources to. Type the path to the folder manually or click the Browse button ... and select the folder in the dialog box, that opens. The data is downloaded recursively, that is, the structure of subfolders under the source node is repeated locally.

Checkout from TFS Wizard: Source Workspace

VCS | Checkout from Version Control | TFS - Choose workspace manually

On this page, specify the workspace to add the imported files and folders to. The data will be downloaded according to the mappings defined in the selected workspace. The data is downloaded recursively, that is, the structure of subfolders under the selected source node is reproduced locally.

You can use an existing workspace as is, or update it as necessary, or even create an entirely new workspace manually.

Item	Description
Server/Workspace	This read-only field shows the URL addresses of TFS servers you have access to and workspaces available on these TFS servers.
Workspace comment	This read-only field shows the descriptions of workspaces on the servers you have access to.
Team Servers	Use the buttons in this area to manage the list of available servers and workspaces and configure access to them. – Add: click this button to open the Add Team Foundation Server dialog box where you can specify the parameters for establishing connection to a TFS server. TFS uses NTLM authentication , so native Windows applications (that is, Microsoft Team Explorer) authenticate silently with system credentials. PhpStorm users must always specify their username and password because of limitations posed by Java Runtime. – Remove: click this button to remove the selected server from the list. – Reload workspaces: click this button to have the list of available workspaces refreshed. – TFS Proxy: click this button to open the Set TFS proxy for server... dialog box where you can specify the parameters for accessing the selected server via Proxy. – Check-in Policies: click this button to open the Edit Check-in Policies dialog box where you can manage the list of check-in policies to be applied.
Workspaces	Use the buttons in this area manage the list of available workspaces and update the workspaces, when applicable. – Create: click this button to open the Create Workspace dialog box for creating a new workspace. – Edit: click this button to open the Edit Workspace dialog box for editing the selected workspace. – Delete: click this button to remove the selected workspace from the list.

Checkout from TFS Wizard: Choose Source Path

VCS | Checkout from Version Control | TFS - Choose workspace manually

The page opens when you choose an existing workspace on the Source Workspace page and click Next.

On this page, specify the folder on the server to download data from.

The page shows a tree of folders on the server that you have access to. When you select the required folder, $product$ tells you which local folder it is mapped to in the current workspace .

Checkout from TFS Wizard: Summary

VCS | Checkout from Version Control | TFS - Create workspace automatically

VCS | Checkout from Version Control | TFS - Choose workspace manually

This last, read-only, page of the wizard shows the name of the workspace to be used or generated, the URL address of the server, the source folder to download data from, and the local folder to save the downloaded data in. Review the details and click Finish.

Commit Changes Dialog

VCS | Commit Changes

View | Tool Windows | Version Control - Local Changes - Context menu of a file or change list - Commit Changes

---

Use this dialog box to commit (check in) changes from the selected changelist to the repository and, optionally, to create a patch file.

This dialog box consists of several areas:

- Modified files pane
  - Toolbar
- Commit Message pane
  - Toolbar
- VCS-specific controls
- Before Submit / Before Commit section
- After Submit / After Commit section
- Details pane
  - Toolbar
- Submit / Commit button

The options available in this dialog depend on the version control system you are using.

Modified files pane

This section contains a list of files that were modified since the last commit. All files in this list are selected by default. Deselect the check-boxes next to the files that you want to exclude from the current commit.

Toolbar

Item	Tooltip and Shortcut	Description	Available in
	Show Differences Ctrl+D	Click this button to open the Differences dialog box that points at the inconsistencies between your local working copy of the selected file and the file in the repository.	All VCSs
	Refresh Changes Ctrl+F5	Click this button to reload the Changed files tree view so it is up-to-date.	All VCSs
	Show Unversioned Files	Click this button if you want to see newly added files that have not been added to version control yet under the Unversioned Files node.	ALl VCSs
	Add to VCS Ctrl+Alt+A	Click this button to move the files selected under the Unversioned Files node to the active changelist, so that they are added to your version control system during the commit.	All VCSs
	Move to Another Changelist F6	Click this button to add the selected file(s) to another changelist. The Move to Another Changelist dialog box opens where you can select an existing changelist or create a new one.	All VCSs
	Delete	Click this button to delete the selected file.	
	Ignore	Click this button to leave the selected files unversioned.	All VCSs
	Rollback	Click this button to revert all changes made to the local working copy of the selected file(s).	All VCSs
	Jump to source F4	Click this button to open the source code of the selected file in the editor.	All VCSs
	Revert Unchanged Files	Click this button to revert the files that have not been modified locally.	Subversion Perforce
	Group by Directory Ctrl+P	Click this button to toggle between the flat view and the directory tree view.	All VCSs
	Expand or collapse all nodes Ctrl+NumPad Plus Ctrl+NumPad -	Click these buttons to expand or collapse all nodes in the directory tree. These buttons are not available in flat view.	All VCSs
Change list		N/A From this drop-down list, select the changelist that contains the modified files to be checked in or included in the patch. The active changelist is selected by default.	All VCSs

The summary under the modified files pane shows statistics on the currently selected changelist, such as the number of modified, new and deleted files. This area also shows how many files of each type are shown, and how many of them will be committed.

Commit Message pane

In this area, enter a comment to the current commit. You cannot commit your changes until you enter some description in the Commit Message field.

This comment will also be used as the name of the patch file, if you decide to create a patch.

Toolbar

Icon and Tooltip	Shortcut	Description
Check commit message spelling	N/A	Click this icon if you want to automatically check the spelling in your commit messages.
Commit Message History	Ctrl+M	Click this icon to invoke the Commit Message History dialog that contains a list of your twenty-five last commits and the corresponding commit messages.

VCS-specific controls

The controls in this section are located in the top-right part of the dialog, and contain the options that are specific for the version control system you are using.

Item	Description	Available for
Author	Use this drop-down list to select the author of the changes that you are going to commit. This may be useful when you are committing changes made by another person.	Git
Amend commit	Select this check box to replace the previous commit with the current changes (see Git Basics: Undoing Things for details).	Git, Mercurial
Keep files locked	Select this check box to keep the changed files locked after they are checked in.	Subversion
Jobs	This area is available only if you select the Enable Perforce Jobs Support check box on the Perforce page of the Settings dialog box. Use the controls in this area to search for Perforce jobs , link jobs to the selected changelist, and detach the currently linked jobs. – Unlink selected jobs: click this button to detach the selected job from the changelist. – Edit associated jobs: click this button to open the Edit Jobs Linked to Changelist dialog where you can search for available jobs, view their details, and link jobs to the selected changelist. – Find and link job matching the pattern: click this button to start quick search for the job that matches the pattern specified in the text box and attach the job to the changelist. In the text box, specify the exact name of the job or a search pattern according to the Perforce jobs syntax rules . Tip If only one job matching the pattern is found, it is attached to the changelist automatically. Otherwise, to select a job among several available jobs, click the button and find the desired job using the Edit Jobs Linked to Changelist dialog box. The list box in the bottom of the area displays the jobs that are currently attached to the selected changelist.	Perforce

Before Submit / Before Commit section

Use the controls in this area to define which additional actions you want PhpStorm to perform before committing the selected files.

These controls are available for the following version control systems:

– Git

– CVS

– Subversion

– Perforce

Item	Description
Reformat code	Select this check box to perform code formatting according to the Project Code Style settings .
Rearrange code	Select this check box to rearrange your code according to the arrangement rules preferences .
Optimize imports	Select this check box to remove redundant import statements .
Perform code analysis	Select this check box to run code inspection on the files you are about to commit.
Check TODO (<filter name>)	Select this check box to review the TODO items matching the specified filter. Click the Configure link to choose an existing TODO filter , or open the TODO settings page and define a new filter to be applied.
Cleanup	Select this check box if you want to automatically apply the current inspection profile to the files you are going to commit.
Revert unchanged files	Select this check box to revert the files that have not been modified. This option is only available for Perforce.

After Submit / After Commit section

Use the controls in this area to define which additional actions you want PhpStorm to perform after committing the selected files.

Item	Description	Available for
Run tool	From this drop-down list, select the external tool that you want PhpStorm to launch after the selected changes have been committed. You can select a tool from the list, or click the Browse button and configure an external tool in the External Tools dialog box that opens.	All VCSs
Upload files to	From this drop-down list, select the server access configuration to use for uploading the committed files to a local or remote host, a mounted disk, or a directory. To suppress uploading, choose None. To add a server configuration to the list, click and fill in the required fields in the Add Server dialog that opens.	All VCSs
Always use selected server	Select this check box to always upload files to the selected server access configuration. The drop-down list and the check box are only available if the Remote Hosts Access plugin is enabled.	All VCSs

Tag committed files	Select this check box to assign a tag to the committed files and type the name of the tag. To replace a previously assigned tag with a new one, select the Override existing tags option.	CVS
Auto-update after commit	Select this check box to automatically update your project after the commit. Enabling this option will help prevent your working copy against the mixed-revision state . The mixed-revision state of a working copy may affect the Move and Rename refactoring applied to folders, in which case items in revisions different from the moved subtree root will be tracked separately, which can be confusing. When the Auto-update after commit option is enabled: – Merge will fail with an error if the merge target is a mixed-revision working copy. – Your own changes will never cause a 409 conflict.	Subversion

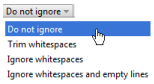
Details pane

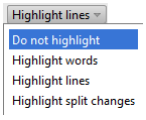
The Details pane is hidden by default. To unfold it, click the arrow button next to the pane title.

In this pane you can explore the differences between the base repository version of the selected file, and the version you are about to commit.

Toolbar

Item	Tooltip and Shortcut	Description
	Previous Difference / Next Difference Shift+F7 F7	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to click the arrow buttons F7 / Shift+F7 once more and compare other files, depending on the Go to the next file after reaching last change option in the Differences Viewer settings . This behavior is supported only when the Differences Viewer is invoked from the Version Control tool window.
	Compare Previous/Next File Alt+Left Alt+Right	Click these buttons to compare the local copy of the previous/next file with its update from the server. Tip These buttons are only available when there is more than one file in the selected changelist.
	Jump to Source F4	Click this button to open the selected file in the active pane in the editor. The caret will be placed in the same position as in the Differences Viewer.
Viewer type		Use this drop-down list to choose the desired viewer type. The side-by-side viewer has two panels; the unified viewer has one panel only. Both types of viewers enable you to – Edit, for example, type text. Note that one can change text only in the right-hand part of the default viewer, or, in case of the unified viewer, in the lower ("after") line. – Perform the Apply/Append/Revert actions.

Whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

Highlighting mode		Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted – Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Collapse unchanged fragments	Click this button to collapse all unchanged fragments in both files. The amount of non-collapsible unchanged lines is configurable in the Diff & Merge settings page.
	Synchronize scrolling	Click this button to scroll both differences panes simultaneously. If this button is released, each pane can be scrolled independently.
	Disable editing	Click this button to enable editing of the local copy of the selected file, which is disabled by default. When editing is enabled, you can make last-minute changes to the modified file before committing it.
	Editor settings	Click this button to open a drop-down list of available options. Select or clear these options to show or hide line numbers, indentation guides, white spaces, and soft wraps.
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools settings page. This button only appears on the toolbar when the Use external diff tool option is enabled in the External Diff Tools settings page.

?	Help F1	Click this button to show the corresponding help page.

Note that the options listed above are available for text files only. PhpStorm cannot compare binary files, so most commands will be unavailable for them.

Submit / Commit button

Click this button to commit the selected files, or hover your mouse over this button to display one of the following available commit options:

- Commit and Push: select this option to push the changes to the remote repository immediately after the commit. This option is available if you are using Git or Mercurial as a version control system.
- Create MQ Patch: select this option to create an MQ patch based on your changes. This option is only available if you are using Mercurial as a version control system.
- Create Patch: select this option if you want PhpStorm to generate a patch based on the changes you are about to commit. In the Create Patch dialog that opens, type the name of the patch file and specify whether you need a reverse patch.
- Remote Run: select this option to run your personal build . This option is only available when you are logged in to TeamCity . Refer to TeamCity plugin documentation for details.

Configure Ignored Files Dialog

Version Control tool window | Local Changes -

Use this page to create a list of files and directories that you do not want to put under version control. These can be file names associated with VCS administration, backup files, and any other artifacts that you want to remain unversioned. You can also specify patterns that define files to ignore.

Item	Keyboard shortcut	Description
+	Alt+Insert	Use this icon or shortcut to add an item to the list. The Ignore Unversioned Files dialog box opens where you can type an exact path to a file or directory to be ignored or specify a pattern that defines the names of files and directories to be ignored.
	Enter	Use this icon or shortcut to edit the selected path or pattern in the Ignore Unversioned Files dialog box.
−	Alt+Delete	Use this icon or shortcut to remove the selected path or pattern from the list.

Create Patch Dialog

VCS | Create Patch

View | Tool Windows | Version Control - Local Changes - Context menu of a file or changelist - Create Patch

Use this dialog box to generate a patch file for the specified changelist or files.

Use this dialog box to create a patch from the selected changelist or files.

This dialog box consists of several areas:

- Modified files pane
  - Toolbar
- Commit Message pane
  - Toolbar
- Details pane
  - Toolbar

Modified files pane

This section contains a list of files that were modified since the last commit. All files in this list are selected by default. Deselect the check-boxes next to the files that you want to exclude from the patch.

Toolbar

Item	Tooltip and Shortcut	Description	Available in
	Show Differences Ctrl+D	Click this button to open the Differences dialog box that points at the inconsistencies between your local working copy of the selected file and the file in the repository.	All VCSs
	Refresh Changes Ctrl+F5	Click this button to reload the Changed files tree view so it is up-to-date.	All VCSs
	Show Unversioned Files	Click this button if you want to see newly added files that have not been added to version control yet under the Unversioned Files node.	ALl VCSs
	Add to VCS Ctrl+Alt+A	Click this button to move the files selected under the Unversioned Files node to the active changelist, so that they are added to your version control system during the commit.	All VCSs
	Move to Another Changelist F6	Click this button to add the selected file(s) to another changelist. The Move to Another Changelist dialog box opens where you can select an existing changelist or create a new one.	All VCSs
	Delete	Click this button to delete the selected file.	
	Ignore	Click this button to leave the selected files unversioned.	All VCSs
	Rollback	Click this button to revert all changes made to the local working copy of the selected file(s).	All VCSs
	Jump to source F4	Click this button to open the source code of the selected file in the editor.	All VCSs
	Revert Unchanged Files	Click this button to revert the files that have not been modified locally.	Subversion Perforce
	Group by Directory Ctrl+P	Click this button to toggle between the flat view and the directory tree view.	All VCSs
	Expand or collapse all nodes Ctrl+NumPad Plus Ctrl+NumPad -	Click these buttons to expand or collapse all nodes in the directory tree. These buttons are not available in flat view.	All VCSs
Change list	N/A	From this drop-down list, select the changelist that contains the modified files to be checked in or included in the patch. The active changelist is selected by default.	All VCSs

The summary under the modified files pane shows statistics on the currently selected changelist, such as the number of modified, new and deleted files. This area also shows how many files of each type are shown, and how many of them will be included in the patch.

Commit Message pane

The comment you enter in this area will be used as the name of the patch file.

Toolbar

Icon and Tooltip	Shortcut	Description
Check commit message spelling	N/A	Click this icon if you want to automatically check the spelling in your commit messages.

Commit Message History	Ctrl+M	Click this icon to invoke the Commit Message History dialog that contains a list of your twenty-five last commits and the corresponding commit messages.

Details pane

The Details pane is hidden by default. To unfold it, click the arrow button next to the pane title.

In this pane you can explore the differences between the base repository version of the selected file, and the version you want to include in the patch.

Toolbar

Item	Tooltip and Shortcut	Description
	Previous Difference / Next Difference Shift+F7 F7	Use these buttons to jump to the next/previous difference. When the last/first difference is hit, PhpStorm suggests to click the arrow buttons F7 / Shift+F7 once more and compare other files, depending on the Go to the next file after reaching last change option in the Differences Viewer settings . This behavior is supported only when the Differences Viewer is invoked from the Version Control tool window.
	Compare Previous/Next File Alt+Left Alt+Right	Click these buttons to compare the local copy of the previous/next file with its update from the server. Tip These buttons are only available when there is more than one file in the selected changelist.
	Jump to Source F4	Click this button to open the selected file in the active pane in the editor. The caret will be placed in the same position as in the Differences Viewer.
Viewer type	Unified viewer Side-by-side viewer Unified viewer	Use this drop-down list to choose the desired viewer type. The side-by-side viewer has two panels; the unified viewer has one panel only. Both types of viewers enable you to – Edit, for example, type text. Note that one can change text only in the right-hand part of the default viewer, or, in case of the unified viewer, in the lower ("after") line. – Perform the Apply/Append/Revert actions.

Whitespace	Do not ignore
	Do not ignore
	Trim whitespaces
	Ignore whitespaces
	Ignore whitespaces and empty lines
		Use this drop-down list to define how the differences viewer should treat white spaces in the text.
		– Do not ignore: white spaces are important, and all differences are highlighted. This option is selected by default.
		– Trim whitespaces: `("\t", " ")` , if they appear in the end and in the beginning of a line.
		  – If two lines differ in trailing whitespaces only, these lines are considered equal.
		  – If two lines are different, such trailing whitespaces are not highlighted in the By word mode.
		– Ignore whitespaces: white spaces are not important, regardless of their location in the source code.
		– Ignore whitespaces and empty lines: the following entities are ignored:
		  – all whitespaces (as in the 'Ignore whitespaces' option)
		  – all added or removed lines consisting of whitespaces only
		  – all changes consisting of splitting or joining lines without changes to non-whitespace parts.
		
		  For example, changing `a b c` to `a \n b c` is not highlighted in this mode.

Highlighting mode	Highlight lines Do not highlight Highlight words Highlight lines Highlight split changes	Select the way differences granularity is highlighted. The available options are: – Highlight words: the modified words are highlighted – Highlight lines: the modified lines are highlighted – Highlight split changes: if this option is selected, big changes are split into smaller 'atomic' changes. For example, `A \n B` vs. `A X \n B X` will be treated as two changes instead of one. – Do not highlight: if this option is selected, the differences are not highlighted at all. This option is intended for significantly modified files, where highlighting only introduces additional difficulties.
	Collapse unchanged fragments	Click this button to collapse all unchanged fragments in both files. The amount of non-collapsible unchanged lines is configurable in the Diff & Merge settings page.
	Synchronize scrolling	Click this button to scroll both differences panes simultaneously. If this button is released, each pane can be scrolled independently.
	Disable editing	Click this button to enable editing of the local copy of the selected file, which is disabled by default. When editing is enabled, you can make last-minute changes to the modified file before committing it.
	Editor settings	Click this button to open a drop-down list of available options. Select or clear these options to show or hide line numbers, indentation guides, white spaces, and soft wraps.
	Show diff in external tool	Click this button to invoke an external differences viewer, specified in the External Diff Tools settings page. This button only appears on the toolbar when the Use external diff tool option is enabled in the External Diff Tools settings page.
	Help F1	Click this button to show the corresponding help page.

Note that the options listed above are available for text files only. PhpStorm cannot compare binary files, so most commands will be unavailable for them.

After you've selected the files you want to commit, click the Create Patch button and specify the patch file options in the dialog that opens.

File Status Highlights

In PhpStorm each file has its own status marked with a specific color. The file status denotes correspondence of the actual file content with the one marked as 'current'.

In the editor, each line in a file is checked whether it corresponds to the state at the 'current' point and marked with a specific color at the left gutter area.

You can customize the default colors:

- For files - in the File Status page of the Colors and Fonts settings.
- For the lines in the editor - in the VCS page of the Colors and Fonts settings.

In this section:

- File Status in Views .
- Line Status in the Editor .

File Status in Views

Color	File Status	Description
Black	Up to date	File is unchanged. test1.txt
Gray	Deleted	File is scheduled for deletion from the repository. privatent.txt
Blue	Modified	File has changed since the last synchronization. test1.txt
Green	Added	File is scheduled for addition to the repository. test1.txt
Violet	Merged	File is merged by your VCS as a result of an update. test1.txt
Brown	Unknown	File exists locally, but is not in the repository, and is not scheduled for adding. Test.xml
Olive	Ignored	File will be ignored in any VCS operation. Test.html
Light brown	Hijacked	File is modified without checkout . This status is valid for the files under Perforce, ClearCase and VSS. HelpTOC.xml
Red	Merged with conflicts	During the last update, file was merged with conflicts. test1.txt
Lilac	Externally deleted	File is deleted locally, but was not scheduled for deletion, and still exists in the CVS repository. test1.txt
Dark cyan	Switched	The file is taken from a different branch than the whole project. This status is valid for CVS and SVN. test1.txt

Line Status in the Editor

Color	File Status	Description
	Modified	Denotes the lines modified since the last synchronization.
	Added	Denotes the lines added since the last synchronization.
	Deleted	Denotes the lines removed since the last synchronization.

New Changelist Dialog

Version Control tool window - +

Alt+Insert

Use this dialog box to create a new changelist.

Item	Description
Name	Type the name of the new changelist.
Comment	Type optional comment. When the new changelist will be submitted to the repository, this comment will appear in the Comment text area of the Commit Changes dialog box.
Make this changelist active	Select this check box to have PhpStorm automatically give the active status to the new change list immediately after the changes are restored in it. When this check box is cleared, the current active changelist remains active. See Changelist for details.
Track context	Select this check box to have PhpStorm preserve the context of the task associated with the new changelist on its deactivation and restore the context when the changelist becomes active. See Managing Tasks and Context for details.

Patch File Settings Dialog

VCS | Create Patch - Create Patch

View | Tool Windows | Version Control - Local Changes - Context menu of a file or changelist - Create Patch - Create Patch

Use this dialog to configure the patch file settings.

Item	Description
Patch file	Specify the name of the patch file. By default, the text in the Commit Message section of the Create Patch dialog is used as the file name. If the Commit Message section is empty, the default name is `unnamed.patch`.
Reverse patch	Select this option if you want to create a patch that reverts the changes you have made.
Include entire base revision content into patch file	Select this option if you want to store the entire base revision text of the modified file inside the patch file. This option is applicable to non-binary files smaller than 500kB.
Encoding	Select the encoding for the patch file from the drop-down list.

Push Dialog (Mercurial, Git)

VCS | Git | Push

VCS | Mercurial | Push

This dialog is available for the following version control systems:

- Git
- Mercurial

The dialog consists of two panes (the Repositories pane and the Commit details pane) and the Push controls area:

- Repositories pane
- Commit details pane
- Push controls

Repositories pane

The left pane shows a list of Git and/or Mercurial repositories (as well as which local branch/active bookmark will be pushed to which remote branch), and a list of commits performed in each repository.

- Hover the mouse over a commit: a tooltip is displayed showing the commit number, date and time, author, and the commit message. If the author of a commit is different from the current user, this commit is marked with an asterisk.
- Select the checkbox next to each repository to which you want to push.
  If you have a multirooted project where repositories are not controlled synchronously, only the current repository is selected by default (or multiple repositories selected in the Project View). For details on how to enable/disable synchronous repositories control, refer to the following sources:
  - for Git: Version Control Settings: Git
  - for Mercurial: Version Control Settings: Mercurial
- To modify the target branch where you want to push (it is highlighted in blue), click it. The label turns into a text field where you can specify the target branch. You can also switch into the editing mode by selecting the branch that you want to modify and pressing `Enter`.
- For Git, you can also edit the remote repository (if there are multiple ones) in the same way as the remote branch.
- If there are no remotes in the repository, the Define remote link appears. Click this link and specify the remote name and URL in the dialog that opens.

Commit details pane

The right pane shows which files are included in the selected commit. If you select multiple branches in the left pane, all corresponding commits will be shown.

The toolbar in this area provides the following options:

Item	Tooltip and shortcut	Description
	Show Diff `Ctrl+D`	Click this button to open the Differences viewer for files dialog that shows the differences between the committed version of the selected file and it previous version.
	Edit Source `F4`	Click this button to open the selected file in the editor.
	Group by Directory `Ctrl+P`	Click this button to toggle between the flat view and the directory view.
	Collapse All / Expand All `Ctrl+NumPad -` / `Ctrl+NumPad Plus`	Click these buttons to fold/unfold all nodes in the directory tree. These buttons are unavailable if the flat view is selected.

Push controls

The controls in this area allow you to select the following push options:

Item	Description
Push Tags	This option is only available if you are using Git. By default, when you perform the `push` operation, tags are not sent to the remote repositories. Select this option if you want to push tags with your commits. – Select All if you want to push all tags, including the tags that do not belong to the selected branches you are about to push (equivalent to `push --tags`). – Select Current Branch if you want to push only the tags that belong to the selected branches you are about to push (equivalent to `push --follow-tags` available since Git 1.8.3).
Export Active Bookmarks	This option is only available if you are using Mercurial. By default, when you perform the `push` operation, bookmarks are not sent to the remote repositories. Select this option if you want to push active bookmarks with your commits.
Push	Click this button and select which operation you want to perform from the drop-down menu: `push` or `push --force`. Select Force push if: – You have created a local branch and want to upload it to the remote repository – You do not want to update the local repository before push For Git, these choice options are only available if the Allow force push option is enabled (see Version Control Settings: Git), otherwise, you can only perform the `push` operation.

Revert Changes Dialog

VCS | Show Changes View - Local - Context menu of a file or change list - Revert Changes

View | Tool Windows | Changes - Local - Context menu of a file or change list - Revert Changes

---

Use this dialog box to roll back changes that have not yet been committed to the repository.

Toolbar

Item	Tooltip and Shortcut	Description
	Show Differences Ctrl+D	Click this button to open the Differences dialog box that points at the inconsistencies between your local working copy of the selected file and the file in the repository.
	Move to Another Changelist F6	Click this button to add the selected file(s) to another changelist. The Choose Changelist dialog box opens where you can select an existing changelist or create a new one.
	Group by Directory Ctrl+P	Click this button to toggle between the flat view and the directory tree view.
	Expand or collapse all nodes Ctrl+NumPad Plus Ctrl+NumPad -	Click these buttons to expand or collapse all nodes in the directory tree. These buttons are not available in flat view.
	Select All Ctrl+A	Click this button to select all the files in the list or directory tree.

Controls

Item	Description
Changed files	This tree view displays the list of changed files. Select check boxes next to the files to be reverted.
Change list	Use the drop-down list to select the change list that contains the modified files to be reverted. By default, the active change list is suggested.
Delete local copies of added files	Use this check box to revert added files as well as your changes in modified ones.

Select Target Changelist Dialog

VCS | Show Changes View - Repository / Incoming - Context menu of a file or change list - Revert Changes

View | Tool Windows | Changes - Repository / Incoming - Context menu of a file or change list - Revert Changes

---

Use this dialog box to roll back changes from a certain change list.

Item	Description
Existing Changelist	Choose this option restore the shelved changes in one of the existing changelists. Choose the target changelist from the drop-down list.
New Changelist	Choose this option to create a new changelist and restore the changes from the shelf in it. – Name: in this text box, type the name of the changelist to be created. – Comment: in this text box, type an optional description of the new changelist. – Make this changelist active: select this check box to have PhpStorm automatically give the active status to the new change list immediately after the changes are restored in it. When this check box is cleared, the current active changelist remains active. See Changelist for details. – Track context: select this check box to have PhpStorm preserve the context of the task associated with the new changelist on its deactivation and restore the context when the changelist becomes active. See Managing Tasks and Context for details.
Remove successfully applied files from the shelf	– Clear this check box to have PhpStorm still display already unshelved changes in the Shelf tab so that you can apply them once more if necessary. – When this check box is selected, the changes are not displayed in the Shelf tab after they are unshelved.

Shelve Changes Dialog

VCS | Shelve Changes

Use this dialog box to put changes in the specified files or changelists to the shelf.

Toolbar

Icon	Tooltip and Shortcut	Description
	Show differences Ctrl+D	Click this button to have differences shown.
	Move to another changelist F6	Click this button to open the Choose Changelist dialog box where you can specify the changelist to move the selected file to.
	Group by Directory Ctrl+P	Use this button to toggle between the flat view and the directory tree view.
	Ctrl+NumPad Plus	Click this button to have all nodes expanded.
	Ctrl+NumPad -	Click this button to have all nodes collapsed.
	Rollback	Click this button to roll the changes back.
	Revert Unchanged Files	Click this button to have unchanged files reverted.
	F4	Click this button to open the source code of the selected file in the editor.
	Message History Ctrl+M	Click this button to reuse one of the previous commit messages (comments) from the list.

Controls

Item	Description
Changed files	This tree view displays the list of changed files. Check the files to be included in the patch.
Comment	Type the comment in the text area. This comment is used as the name of the new patch file.
Changelist	Use the drop-down list to select the changelist that contains the modified files to be included in the patch. By default, the active changelist is suggested.
Summary	This section displays summary information for the currently selected changelist (the number modified, new and deleted files).
Before commit	Use this section to define the following options:
Optimize imports	Check this option to remove redundant import statements.
Reformat code	Check this option to perform code formatting according to the Project Code Style settings .
Perform code analysis for affected files	Check this option to run code inspection on the affected files.
Shelve changes	Click this button to perform shelving and close the dialog.

Show History for File / Selection Dialog

VCS | Local History | Show History or Show History for Selection

Use this dialog to explore changes to a file, or selection. There are two views in this dialog:

- History view in the left-hand part
- Differences view in the right-hand part

Tip The same dialog boxes are available on the context menu of a file or selected text in the editor.

History view

This view shows the list of revisions (states) of a file, with the date and time when the revision was stored. Some of the revisions are supplied with tags and labels.

Revisions are tagged automatically, for example, on opening a project, committing changes, or performing test. You can also set your own labels .

Item	Description
	Click this button to revert the selected action.
	Click this button to create a patch based on the selected local version.
	Click this button to open the corresponding help topic.

Tip The same actions are available on the context menu of each revision.

Differences view

The Differences view is a powerful editor that supports basic search and replace , undo/redo actions , and code completion .

If a revision is selected in the History view , the left-hand pane of the Differences view shows this read-only revision, with the differences against the current revision which is displayed in the right-hand pane. The current revision can be edited.

Item	Shortcut	Description
	Ctrl+C	Click this button to copy the current line or the selected fragment to the clipbord.
or	F7 or Shift+F7	Use these buttons to move to the next or previous difference.
Ignore whitespace		Use this drop-down list to define how the differences viewer should treat white spaces in the text. – Do not ignore - when this option is selected, white spaces are considered unimportant and the differences are highlighted. – Leading and Trailing - select this option to have differences in the end and in the beginning of a line ignored. – All - when this option is selected, white spaces are considered unimportant regardless of their location in the source code.
, or		Use these buttons to apply differences.
Legend		This area shows summary information about the encountered differences: the number of differences found and the color map. The color map for the Differences viewer is configured on the Colors and Fonts page .

Show History for Folder Dialog

VCS | Local History | Show History

Use this dialog to explore changes to a folder. There are two views in this dialog:

- History view in the left-hand part
- Changes view in the right-hand part

Tip The same dialog box is available on the context menu of a folder.

History view

The History view shows a list of folder revisions (states), each one being supplied with a time stamp, revision number, and an indication of the action that resulted in that state.

Item	Description
	Click this button to revert the selected action.
	Click this button to create a patch based on the selected local version.
	Click this button to open the corresponding help topic.

Changes view

The Changes view shows the differences between the current state and the one selected in the History view . The differences are shown as a tree of changed (new, modified and deleted) files and subfolders.

Item	Shortcut	Description
	Ctrl+D	Click this button to show the differences between the current local version and the one selected in the History view. Alternatively, double-click the current local version in the Changes view.
		With a file selected in the Changes view, click this button to roll back the selected action.
	Ctrl+P	Click this button to show changed files as a tree view of folders. If this button is not pressed, the files are shown as a flat list.
or	Ctrl+NumPad Plus or Ctrl+NumPad -	Click these buttons to have all nodes expanded or collapsed. These buttons are only available when the changed files are shown as a tree view.
	Ctrl+A	Click this button to select all the files in the list or a tree view.
	Ctrl+F	In this area, type the search string. Note also that speed search is available in the Changes pane.
		Click this button to clear the search area.

Tip Speed search is available in the Version Control view.

Unshelve Changes Dialog

Version Control Tool window | context menu | Unshelve Changes

Use this dialog box to restore shelved changes from a shelf to a changelist.

Item	Description
Existing Changelist	Choose this option restore the shelved changes in one of the existing changelists. Choose the target changelist from the drop-down list.
New Changelist	Choose this option to create a new changelist and restore the changes from the shelf in it. – Name: in this text box, type the name of the changelist to be created. – Comment: in this text box, type an optional description of the new changelist. – Make this changelist active: select this check box to have PhpStorm automatically give the active status to the new change list immediately after the changes are restored in it. When this check box is cleared, the current active changelist remains active. See Changelist for details. – Track context: select this check box to have PhpStorm preserve the context of the task associated with the new changelist on its deactivation and restore the context when the changelist becomes active. See Managing Tasks and Context for details.
Remove successfully applied files from the shelf	– Clear this check box to have PhpStorm still display already unshelved changes in the Shelf tab so that you can apply them once more if necessary. – When this check box is selected, the changes are not displayed in the Shelf tab after they are unshelved.

Table Editor

From the Database tool window (for any table within a DB data source):

- on the toolbar (if the toolbar is not currently hidden)
- Table Editor from the context menu
- F4

Use the Table Editor to manipulate the table data, and to perform other, associated tasks.

There are two tabs in the Table Editor:

- Data. On this tab, the table you are working with is shown.
- DDL. Initially, the `CREATE TABLE` statement used to create the table is shown on this tab. You can edit the statement and then run it (on the toolbar or Ctrl+Shift+F10). Use or Ctrl+F5 to regenerate the `CREATE TABLE` statement for the current state of the table in the database.

Below, the features of the Data tab are discussed.

- Toolbar controls, context menu commands for data cells and keyboard shortcuts
- Using the table header row: sorting data, reordering and hiding columns

See also, Working with the Table Editor .

Toolbar controls, context menu commands for data cells and keyboard shortcuts

Most of the functions on the Data tab are accessed by means of controls on the toolbar, context menu commands for the data cells, and associated keyboard shortcuts.

Item	Shortcut	Description
and		These icons and corresponding commands are for switching between the result set pages, i.e. the pages that show the table data. A fixed number of rows shown simultaneously is referred to as a result set page. If this number is less than the number of rows in the table, only a subset of all the rows is shown at a time. In such cases, you can use and to switch between the subsets. (If all the rows are currently shown, these icons and the corresponding commands are inactive.) The result set page size is set on the Database page of the Settings dialog.
First Page		Use this icon or command to switch to the first of the result set pages to see the first series of rows.
Previous Page	Ctrl+Alt+Up	Use this icon, command or shortcut to switch to the previous result set page to see the previous series of rows.
Next Page	Ctrl+Alt+Down	Use this icon, command or shortcut to switch to the next result set page to see the next series of rows.
Last Page		Use this icon or command to switch to the last of the result set pages to see the last series of rows.
Reload Page	Ctrl+F5	Use this icon, command or shortcut to refresh the current table view. Use this function to: – Synchronize the data shown with the actual contents of the database. – Apply the Result set page size setting after its change.
Add New Row	Alt+Insert	Use this icon, command or shortcut to add a new row to the table. Complete entering a value into a cell by pressing Enter . To save the new row, select Submit New Row from the context menu or press Ctrl+Enter . See also, Adding a row.
Delete Selected Rows	Ctrl+Y	Use this icon, command or shortcut to delete the selected row or rows. Rows are selected by clicking the cells in the column where the row numbers are shown. To select more than one row, use mouse clicks in combination with the Ctrl key.
Auto-commit		Use this check box to turn the autocommit mode for the database connection on or off. In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back. (Each change of a value in the table, or adding or deleting a row results in executing an SQL statement.) If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the commit or rollback command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements. In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.
Commit		If the autocommit mode is off, use this icon or command to commit the current transaction.
Rollback		If the autocommit mode is off, use this icon or command to roll back the current transaction.
Cancel Query	Ctrl+F2	Use this icon or shortcut to terminate execution of the current query.
Tab S...(TSV) Data Extractor		Use this button or command to open a menu in which you can select an output format for your data. The Configure CSV Formats command in that menu opens the CSV Formats Dialog that lets you manage the delimiter-separated values formats (e.g. CSV, TSV).
View Query		Use this button to view the query which was used to generate the current table view. To close the pane where the query is shown, press Escape .
		This icon provides access to the following commands: – Transpose. Use this command to turn the transposed table view on or off. (In the transposed view, the rows and columns are interchanged. So, the rows are shown as columns and vice versa.) – Reset View. Use this command to restore the initial table view after reordering or hiding the columns, or sorting the data. – Sort via ORDER BY. Use this command to turn the corresponding option on or off. If the Sort via ORDER BY option is on, all the sorting operations that you perform are reflected in the corresponding `SELECT` statement

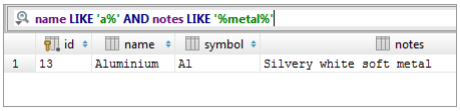
(an `ORDER BY` clause is added or modified) which is executed immediately. As a result, the data for the whole table is sorted by the corresponding database system.

Don't turn this option on if you want to keep interactions with the database to a minimum (e.g. when the table is very big or the database connection is "slow").

If this option is off, the data is sorted "locally" by PhpStorm and only for the rows currently shown.

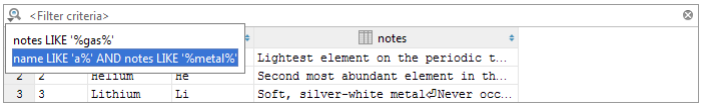
- Row Filter. Use this command to show or hide the box in which you can specify filtering conditions for your table.
- Settings. Use this command to open the Database page of the Settings dialog to view or edit the settings for the Database console, the Table Editor and the Database tool window.

In this box, specify filtering conditions for the table. (If the box is not currently shown, click on the toolbar and select Row Filter.)
The filtering conditions are specified as in a `WHERE` clause but without the word `WHERE`, e. g. `name LIKE 'a%' AND notes LIKE '%metal%'`. Within the `LIKE` expressions, the SQL wildcards can be used: the percent sign (`%`) for zero or more characters and underscore (`_`) for a single character.



To apply the conditions currently specified in the box, press `Enter`. To cancel filtering, delete the contents of the filter box and press `Enter`.

To reapply a memorized filter, click and select the filter in the list. See also, Filter history size .



To cancel filtering and hide the filter box, click .

Clone Row	`Ctrl+D`	Use this command or shortcut to create a copy of the selected row.
Edit Value	`F2`	Use this command or shortcut to start editing a value in the selected table cell. (Alternatively, you can double-click the cell or simply start typing.) To save the modified value, press `Enter`. To cancel editing, press `Escape`. See also, Modifying cell contents.
Quick Documentation	`Ctrl+Q`	Use this command or shortcut to open the quick documentation view. To close the view, press `Escape`. For more information, see Using the quick documentation view.
Go To | Row	`Ctrl+G`	Use this command or shortcut to switch to a specified row. In the dialog that opens, specify the row number to go to.
Go To | Related Data	`F4`	Use this command or shortcut to switch to a related record. The command options are a combination of those for Go To | Referenced Data and Go To | Referencing Data . The command is not available if there are no related records.
Go To | Referenced Data	`Ctrl+B`	Use this command or shortcut to switch to a record that the current record references. If more than one record is referenced, select the target record in the pop-up that appears. The command is not available if there are no referenced records.
Go To | Referencing Data	`Alt+F7`	Use this command or shortcut to see the records that reference the current record. In the pop-up that appears there are two categories for the target records: – First Referencing Row. All the rows in the corresponding table will be shown and the first of the rows that references the current row will be selected. – All Referencing Rows. Only the rows that reference the current row will be shown. The command is not available if there are no records that reference the current one.
Filter by		Use this command to access quick filtering options. The options include those for the current column name and depend on the value in the current cell.
Dump Data | To Clipboard		Use this command to copy the table data onto the clipboard.
Dump Data | To File		Use this command to save the table data in a file. In the dialog that opens, specify the location and name of the file.
Save LOB		Use this command to save the large object (LOB) currently selected in the table in a file.

Using the table header row: sorting data, reordering and hiding columns

Use the cells in the header row (i.e. the row where column names are shown) for:

- Sorting data
- Reordering columns
- Hiding and showing columns

You can sort table data by any of the columns by clicking the cells in the header row.

Each cell in this row has a sorting marker in the right-hand part and, initially, a cell may look something like this: name . The sorting marker in this case indicates that the data is not sorted by this column.

If you click the cell once, the data is sorted by the corresponding column in the ascending order. This is indicated by the sorting marker appearance: name 1 . The number to the right of the marker (1 on the picture) is the sorting level. (You can sort by more than one column. In such cases, different columns will have different sorting levels.)

When you click the cell for the second time, the data is sorted in the descending order. Here is how the sorting marker indicates this order: name ▼1.

Finally, when you click the cell for the third time, the initial state is resorted. That is, sorting by the corresponding column is canceled: name ⇕.

Here is an example of a table where data are sorted by two of its columns.

	id	name ▲2	relation ▲1
1	3	Dylan	brother
2	6	Jack	brother
3	1	Harry	father
4	2	Chloe	mother
5	5	Alice	sister
6	4	Emily	sister

To restore the initial "unsorted" state for the table, click ⚙ and select Reset View. See also, Sort via ORDER BY.

To reorder columns, use drag-and-drop for the corresponding cells in the header row. To restore the initial order of columns, click ⚙ and select Reset View.

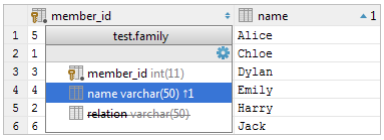
	member_id	relation	name ▲1
1	5	sister	lice
2	1	mother	hloe
3	3	brother	ylan
4	4	sister	mily
5	2	father	arry

To hide a column, right-click the corresponding header cell and select Hide column.

To show a hidden column:

1. Do one of the following:
   - Right-click any of the cells in the header row and select Column List.
   - Press `Ctrl+F12`.

   In the list that appears, the names of hidden columns are shown struck through.



2. Select (highlight) the column name of interest and press `Space`.
3. Press `Enter` or `Escape` to close the list.

To show all the columns, click ⚙ and select Reset View.

See also, Using the Structure view to sort data, and hide and show columns.

Diagram Reference

In this section:

- Diagram Toolbar and Context Menu
- Diagram Preview
- General Techniques of Using Diagrams
- Class Diagram Toolbar and Context Menu

Diagram Toolbar and Context Menu

In this section:

- Toolbar
- Context menu

Toolbar

Item	Description
	Click this button to increase the scale of the diagram. Alternatively, press `NumPad+`.
	Click this button to decrease the scale of the diagram. Alternatively, press `NumPad-`.
	Click this button to restore the actual size of the diagram.
	Click this button to make the contents fit into the current diagram size.
	Click this button to apply the current layout, selected on the context menu of the diagram.
	Click this button to save the current diagram in the specified location as `xml` file.
	Click this button to save the diagram in an image file with the specified name and path. The possible formats are: `jpeg`, `png`, `svg`, `svgz`, or `gif`.
	Click this button to print the diagram.
	Click this button to open the diagram preview in a separate frame, where you can configure the page layout, scale, and headings information.

Context menu

The table below contains commands that are not available from the toolbar.

Item	Description
New	Use this node to add new elements to a diagram.
Refactor	This node contains refactoring commands, enabled in the current context.
Jump to Source	Choose this command to open the selected diagram node element in the editor.
Find Usages	Choose this command to search for usages of the selected node element.
Layout	Select the desired diagram layout from the submenu.
Show Edge Labels	Check this command to show multiplicities in diagram.

Diagram Preview

Use the Structure view as a Preview, that allows you to get a "10,000-feet" look on a diagram. The shadow area represents the visible part of a diagram. As you zoom in or out, or change the shape of the PhpStorm windows, the size of the shadow area changes accordingly.

To enable the diagram preview

- Open the Structure tool window.

In the Preview pane, the following actions are available:

- Keeping the mouse button pressed, move the shadow area to obtain the desired view.
- Select one or more nodes in the diagram, and the corresponding nodes in the Preview are marked dark gray.

General Techniques of Using Diagrams

In this section:

- Selecting elements in diagram .
- Managing diagram layout .
- Zooming in and out .
- Using the magnifier tool .
- Navigating to source code .
- Invoking refactoring commands .
- Finding usages of the selected node element .

To select elements in diagram

- To select an element, just click it in diagram.
- To select multiple adjacent elements, keep `Shift` pressed and click the desired elements, or just drag a *lasso* around the elements to be selected.
- To select multiple non-adjacent elements, keep `Ctrl+Shift` pressed and click the desired elements.
- To select a member of a node element, double-click the node element, and then use the arrow keys, or the mouse pointer.

To manage diagram layout

- Right-click the diagram background, and choose Layout command of the diagram context menu. Next, select the desired layout from the submenu.
- Use Drag-and-drop technique to lay out entities in diagram manually.
- Apply the current layout selected on the context menu of the diagram, by clicking .

To zoom in and out, do one of the following

- Use the and toolbar buttons.
- Keeping `Ctrl` key pressed, rotate your mouse wheel up or down.
- Press `NumPad+` or `NumPad-`.

> **Tip** As you zoom in or out, the size of the shadow area in the diagram preview changes accordingly.

To use the magnifier tool

- Keep the `Alt` key pressed, and hover your mouse pointer over the most interesting, or problematic areas of the diagram.

To jump from an element in diagram to the underlying source code

1. Select an element in diagram.
2. Do one of the following:
   - On the context menu of the diagram, choose Jump to Source
   - Press `F4`.
   - Double-click selected element.

   The source code of the corresponding source file opens in a separate tab in the editor.

Class Diagram Toolbar and Context Menu

In this section:

– Toolbar

– Context Menu

Toolbar

Item	Description
	Click this button to show methods in the class nodes.
	Click this button to show fields in the class nodes.
	Click this button to show constructors in the class nodes.
	Click this button to enable creating extends or implements links between node elements. If this button is not pressed, links cannot be drawn.
	Click this button to increase the scale of the diagram, or press `NumPad+`.
	Click this button to decrease the scale of the diagram, or press `NumPad-`.
	Click this button to restore the actual size of the diagram.
	Click this button to make the contents fit into the current diagram size.
	Click this button to apply the current layout, selected on the context menu of the diagram, or press `F5`.
	Click this button to save the current diagram as a `*.uml` file.
	Click this button to save the diagram in an image file with the specified name and path. The possible formats are: `jpeg`, `png`, `svg`, `svgz`, or `gif`.
	Click this button to print the diagram.
	Click this button to open the diagram preview in a separate frame, where you can configure the page layout, scale, and headings information.

Context Menu

This section describes only those context menu commands that are not available from the toolbar.

Item	Shortcut	Description
Add class to diagram	`Space`	Choose this command to add existing class to the diagram background.
Collapse nodes	`C`	Choose this command to show the containing package of the selected node.
Expand nodes	`E`	Choose this command to show class diagram of the selected package.
New	`Alt+Insert`	Choose this command to create a new node element or member .
Refactor		Point to this node to select one of the refactoring commands available in this context.
Analyze		Point to this node to select one of the code analysis commands available in this context.

Add New Field or Constant

UML Class diagram - Context menu of a node element | New | Field

`Alt+Insert`

Item	Description
Visibility	From this drop-down list, select the desired visibility modifier.
Type	Specify the type of the new field. Note that code completion works here. As you type, the suggestion list shrinks to show the matching options. Non-existent type is marked as a syntax error.
Name	Specify the name of the new field.
Initializer	Specify the initial value of the new field. Note that such value should correspond to the declared type. For example, if the type is String, the suggested initializer should be enclosed in quotes. Note that code completion works here. As you type, the suggestion list shrinks to show the matching options. Non-existent type is marked as a syntax error.
Static	Select this check box to add `static` to the field declaration.
Final	Select this check box to add `final` to the field declaration.
Preview	This read-only area shows the draft declaration as you fill in the dialog fields.

Create New Method

UML Class diagram - Context menu of a node element | New | Constructor

`Alt+Insert`

Item	Description
Name	Use this field to modify the method name.
Return type	Use this field to modify the method return type. Code completion (`Ctrl+Space`) is available in this field, in the Type column of the Parameters area, and in the exception type list in the Exceptions area.
Parameters	
Type	Specify the parameter type. Note that code completion works here. As you type, the suggestion list shrinks to show the matching options. Non-existent type is marked as a syntax error.
Name	Specify the parameter name.
Add/Remove	Use these buttons to add/remove parameters.
Move Up/Down	Use these buttons to reorder parameters by moving them up or down in the parameter list.
Visibility	Select one of the options to change visibility scope of the method.
Abstract	If this check box is selected, the `abstract` modifier is added to the method signature.
Exceptions	
Add/Remove	Use these buttons to add/remove exceptions thrown by the method. For each new exception, specify its type.
Move Up/Down	Use these buttons to reorder exceptions by moving them up or down in the list.
Signature Preview	This read-only area shows the draft method signature as you fill in the dialog fields.

Icons Reference

- File Types Recognized by PhpStorm
- Symbols

File Types Recognized by PhpStorm

PhpStorm recognizes numerous file types. Each file type is denoted with a special icon. Custom files types are also allowed. Each file type is associated with one or more extensions that match a certain pattern.

The file types and their extensions are configurable in the File Types dialog.

Note The file recognized types depend on the installed plugins.

The default types include:

File Type	Icon
Apache Config files	
Archive files	
C# files	
C/C++ files	
Command Shell files	
CSS files	
CoffeeScript files	
Dart files	
Diagram files	
Erlang files	
Files marked as plain text	
Files opened in associated applications	
Handlebars files	
HAML files	
HTML files	
IDL files	
Image files	
JavaFX files	
JavaScript files	
JavaScript test files	
JavaScript files that can be executed on Node.js	
Jade files (refer to the section Using Pug (Jade) Template Engine).	
JSHint configuration files	
JSON files	
JSTestDriver Config files	
Java Server Pages files	
Less files	
reStructuredText files	
Patch files	
Perl files	
PHP files	
Pug files (refer to the section Using Pug (Jade) Template Engine).	
Regular expressions	
RELAX NG Compact Syntax	
Sass files	
SCSS files	
Smarty, Smarty config files	
SQL files	
Stylus files	
Text files	
TypeScript files	
XHTML files	
XML DTD files	
XML files	
YAML files	

Symbols

In this section:

- Icons in the editor, tool windows, dialogs, and in the tree view
- Data Sources

Icons in the editor, tool windows, dialogs, and in the tree view

Icon	Description
	Class
	Final class
	Abstract class
	Interface
	PHP trait
	Test case
	Method/function
	Method/function in an interface
	Variable
	Constant
	Field
	Property
	Parameter
	Element
	Directory
	Source root
	Test root
	Resource root
	Excluded root
	Exception
Visibility modifiers	
	Private
	Protected
	Public

Data Sources

Icon	Description
	DB data source. Also, DBMS-specific icons are used: DB2 Derby H2 HSQLDB MySQL Oracle PostgreSQL SQL Server SQLite Sybase
	DB data source with the read-only status, e.g. for DB2.
	DDL data source
	Schema
	Table
	View
	Column
	A `NOT NULL` column
	Column with a primary key
	Column with a foreign key
	Column with an index

	Primary key
	Foreign key
	Index
	Stored procedure or function

Regular Expression Syntax Reference

This section provides a brief summary of regexp syntax that can be helpful for creating search and issue navigation patterns.

In this section:

- RegEx Syntax Reference
- Tips and Tricks

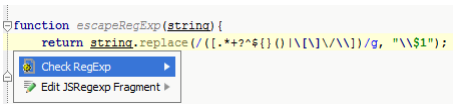
RegEx Syntax Reference

Character	Description
`\`	Marks the next character as either a special character or a literal. For example: – *n* matches the character *n*. "\n" matches a newline character. – The sequence `\\` matches `\` and `\(` matches `(`.
`^`	Matches the beginning of input.
`$`	Matches the end of input.
`*`	Matches the preceding character zero or more times. For example, "zo*" matches either *z* or *zoo*.
`+`	Matches the preceding character one or more times. For example, "zo+" matches *zoo* but not *z*.
`?`	Matches the preceding character zero or one time. For example, `a?ve?` matches the `ve` in `never`.
`.`	Matches any single character except a newline character.
`(` subexpression `)`	Matches *subexpression* and remembers the match. If a part of a regular expression is enclosed in parentheses, that part of the regular expression is grouped together. Thus a regex operator can be applied to the entire group. – If you need to use the matched substring within the same regular expression, you can retrieve it using the backreference (`\num`, where `num = 1..n`). – If you need to refer the matched substring somewhere outside the current regular expression (for example, in another regular expression as a replacement string), you can retrieve it using the dollar sign (`$num`, where `num = 1..n`). – If you need to include the parentheses characters into a *subexpression*, use `\(` or `\)`.
x `|` y	Matches either *x* or *y*. For example, `z|wood` matches *z* or *wood*. `(z|w)oo` matches *zoo* or *wood*.
{ `n` }	*n* is a nonnegative integer. Matches exactly *n* times. For example, `o{2}` does not match the *o* in *Bob*, but matches the first two o's in *foooood*.
{ `n` ,}	*n* is a nonnegative integer. Matches at least *n* times. For example, `o{2,}` does not match the *o* in *Bob* and matches all the o's in "foooood." `o{1,}` is equivalent to `o+`. `o{0,}` is equivalent to `o*`.
`{` *n* `,` *m* `}`	*m* and *n* are nonnegative integers. Matches at least *n* and at most *m* times. For example, `o{1,3}` matches the first three o's in "fooooood." `o{0,1}` is equivalent to `o?`.
`[` *xyz* `]`	A character set. Matches any one of the enclosed characters. For example, `[abc]` matches the *a* in *plain*.
`[^` *xyz* `]`	A negative character set. Matches any character not enclosed. For example, `[^abc]` matches the *p* in *plain*.
`[` *a-z* `]`	A range of characters. Matches any character in the specified range. For example, "[a-z]" matches any lowercase alphabetic character in the range *a* through *z*.
`[^` *m-z* `]`	A negative range characters. Matches any character not in the specified range. For example, `[m-z]` matches any character not in the range *m* through *z*.
`\b`	Matches a word boundary, that is, the position between a word and a space. For example, `er\b` matches the *er* in *never* but not the *er* in *verb*.
`\B`	Matches a non-word boundary. `ea*r\B` matches the *ear* in *never early*.
`\d`	Matches a digit character. Equivalent to `[0-9]`.
`\D`	Matches a non-digit character. Equivalent to `[^0-9]`.
`\f`	Matches a form-feed character.
`\n`	Matches a newline character.
`\r`	Matches a carriage return character.
`\s`	Matches any white space including space, tab, form-feed, etc. Equivalent to `[ \f\n\r\t\v]`.
`\S`	Matches any nonwhite space character. Equivalent to `[^ \f\n\r\t\v]`.
`\t`	Matches a tab character.
`\v`	Matches a vertical tab character.
`\w`	Matches any word character including underscore. Equivalent to `[A-Za-z0-9_]`.
`\W`	Matches any non-word character. Equivalent to `[^A-Za-z0-9_]`.
`\` num	Matches *num*, where *num* is a positive integer, denoting a reference back to remembered matches. For example, `(.)\1` matches two consecutive identical characters.
`\` *n*	Matches *n*, where *n* is an octal escape value. Octal escape values should be 1, 2, or 3 digits long. For example, `\11` and `\011` both match a tab character. `\0011` is the equivalent of `\001` & *1*. Octal escape values should not exceed 256. If they do, only the first two digits comprise the expression. Allows ASCII codes to be used in regular expressions.
`\x` *n*	Matches *n*, where *n* is a hexadecimal escape value. Hexadecimal escape values must be exactly two digits long.

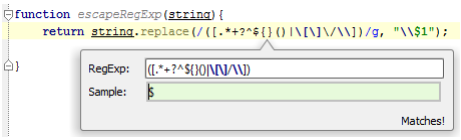
	For example, `\x41` matches `A`. `\x041` is equivalent to `\x04` & `1`. Allows ASCII codes to be used in regular expressions.
`\\$`	Escapes `$`.
`\l`	Changes the case of the next character to the lower case.
`\u`	Changes the case of the next character to the upper case.
`\L`	Changes the case of all the subsequent characters up to `\E` to the lower case.
`\U`	Changes the case of all the subsequent characters up to `\E` to the upper case.

Tips and Tricks

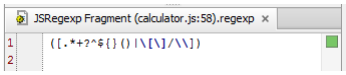
PhpStorm provides intention actions to check validity of the regular expressions, and edit regular expressions in a scratchpad. Place the caret at a regular expression, and press `Alt+Enter`. The suggestion list of intention actions, available in this context, appears:



- Choose Check RegExp, and press `Enter`. The dialog box that pops up, shows the current regular expression in the upper pane. In the lower pane, type the string to which this expression should match. If the regular expression matches the entered string, the background becomes green. If the regular expression doesn't match, then the background is red.



- Choose Edit RegExp Fragment, and press `Enter`. The regular expression opens for editing in a separate tab in the editor. However, this is but a scratchpad, and no file is physically created:



As you type in the scratchpad, all changes are synchronized with the original regular expression. Press `Escape` to close the editor tab.

Scope Language Syntax Reference

The scopes language is used in specifying project scopes involved in various kinds of analysis.

In this section:

- Sets of Files
- Logical Operators
- Defining Scopes
- Examples

Sets of Files

- Single file is defined by a file name, i.e. `MyDir/MyFile.txt`
- Set of all files in a directory, not recursing into subdirectories, is defined by an asterisk after slash, for example: `file:src/main/myDir/*`
- Set of all files in a directory including contents of subdirectories, is defined by an asterisk after double slash, for example `file:src/main/myDir//*`

Logical Operators

The scope language allows you to use common logical operators:

```
&& for AND
                    || for OR
                    ! for NOT
```

Besides that, the parentheses can be used to join the logical operators into groups. For example, the following scope

```
                    (<a>||<b>)&&<c>
```

implies either <a> and <c>, or <b> and <c>.

Defining Scopes

Scopes are defined in the Scopes dialog box in the following ways:

- Manually
- With the pointing device

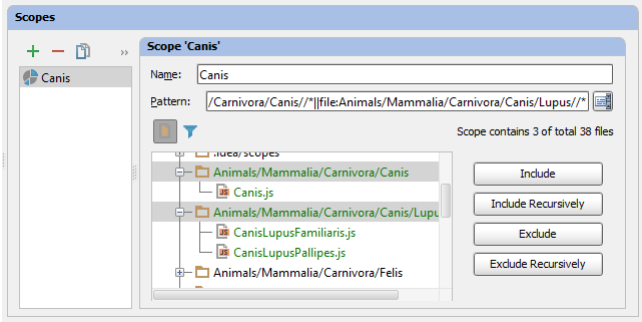
Manually

Specify file masks in the Pattern text box, or click and type the pattern in the editor.

Using the Mouse Pointer

Select files and folders in the project tree view and click the buttons Include, Include Recursively, Exclude, and Exclude Recursively. For information about the controls, refer to Scope page description.

Based on the inclusion/exclusion of file and directories, PhpStorm creates an expression and displays it in the Pattern field.



Examples

- `file:*.php||file:*.twig` - include all PHP and Twig files.
- `file^*php&&!file:*test*.php` - include all PHP files except those that contain the character string `test` in their names.
- `file:www/wp-content/themes/my-site-child//*.php||file:www/wp-content/themes/my-site-child//*.css||file:www/wp-content/themes/my-site-child//*.js||file:www/wp-content/plugins/my-site//*.php||file:www/wp-content/plugins/my-site//*.css||file:www/wp-content/plugins/my-site//*.js` - include all .PHP, .CSS, and .JS files from the theme and plugin of the WordPress website.
- `file:*.js||file:*.coffee` - include all JavaScript and CoffeeScript files.
- `file:*js&&!file:*.min.*` - include all JavaScript files except those that were generated through minification, which is indicated by the `min` extension.

Importing PhpStorm Settings on First Launch

While working with PhpStorm, you've definitely customized settings (for example, editor behavior, or keyboard shortcuts) to fit your needs. Having installed and launched a newer version of PhpStorm (when the `config` directory doesn't yet exist for the new version), you might want to re-use the settings of the previous versions.

For this purpose, PhpStorm automatically shows a dialog box with the following options:

- I want to import settings from the previous version: If this option is selected, PhpStorm automatically detects the home directory of the previous version, and imports settings from this location.
- I want to import settings from custom location: If this option is selected, you can specify the desired directory where the settings are stored.
  Type the path manually, or click the browse button to select settings from the previous version using the file chooser dialog.
- I don't have a previous version of PhpStorm, or I don't want to import settings: If this option is selected, then all the previous settings (if any) will be ignored.

Click the radio-button of the desired option to re-use or ignore the legacy settings.

Note If automatic import fails, consider importing and exporting settings manually, using the File | Export Settings and File | Import Settings commands on the main menu. Refer to section Exporting and Importing Settings .

Project and IDE Settings

In this topic:

- Overview
- Project Settings
- IDE Settings
- Locations of directories

Overview

There are two types of settings that define your preferred environment:

- Project Settings, that apply to a specific project. They are marked with in the Settings/Preferences dialog.
- IDE Settings, that are common for all projects and refer to the project-independent aspects.

Project Settings

Project settings are stored with each specific project as a set of `xml` files under the `.idea` folder. If you specify the default project settings , these settings will be automatically used for each newly created project.

The settings that pertain to a project, are marked with the icon in the Settings/Preferences dialog.

IDE Settings

IDE settings are stored in the dedicated directories under PhpStorm home directory. The PhpStorm directory name is composed of the product name and version.

For example:

Windows

- `<User home>\.PhpStormXX\config` that contains user-specific settings.
- `<User home>\.PhpStormXX\system` that stores PhpStorm data caches.

`<User home>` in WindowsXP is `C:\Documents and Settings\<User name>\` ; in Windows Vista it is `C:\Users\<User name>\`

Linux

- `/.PhpStormXX/config` that contains user-specific settings.
- `~/.PhpStormXX/system` that stores PhpStorm data caches.

OS X

- `~/Library/Application Support/PhpStormXX` contains the catalog with plugins.
- `~/Library/Preferences/PhpStormXX` contains the rest of the configuration settings.
- `~/Library/Caches/PhpStormXX` contains data caches, logs, local history, etc. These files can be quite significant in size.
- `~/Library/Logs/PhpStormXX` contains logs.

Locations of directories

The `config` directory has several subfolders that contain xml files with your personal settings. You can easily share your preferred keymaps, color schemes, etc. by copying these files into the corresponding folders on another PhpStorm installation. Prior to copying, make sure that PhpStorm is not running, because it can erase the newly transferred files before shutting down.

The following is the list of some of the subfolders under the `config` folder, and the settings contained therein.

Folder name	User Settings
`codestyles`	Contains code style schemes .
`colors`	Contains editor colors and fonts customization schemes.
`filetypes`	Contains user-defined file types .
`inspection`	Contains code inspection profiles .
`keymaps`	Contains PhpStorm keyboard shortcuts customizations.
`options`	Contains various options, for example, feature usage statistics and macros.
`templates`	Contains user-defined live templates .
`tools`	Contains configuration files for the user-defined external tools .
`shelf`	Contains shelved changes .

Locations of the `config` , `system` , `plugins` directories can be modified in `idea.properties` file.

PhpStorm makes it possible to change `*.vmoptions` and `idea.properties` files without editing them in the PhpStorm installation folder.

To create an empty `idea.properties` file or to copy `*.vmoptions` file, choose the Help | Edit Custom Properties or Help | Edit Custom VM Options on the main menu respectively. Refer to the menu items description for details.

To learn how to change the `idea.properties` file, read the section File 'idea.properties' .

Directories Used by PhpStorm to Store Settings, Caches, Plugins and Logs

Location of the IDE files depends on the operating system, and PhpStorm version.

In this section:

- Windows
- Linux and the other UNIX systems
- OS X

Windows

All the files are located under this directory by default:

Windows Vista, 7, 8, 10

```
<SYSTEM DRIVE>\Users\<USER ACCOUNT NAME>\.<PRODUCT><VERSION>
```

Windows XP

```
<SYSTEM DRIVE>\Documents and Settings\<USER ACCOUNT NAME>\.<PRODUCT><VERSION>
```

Example

- PhpStorm 2016.1

```
c:\Users\John\.PhpStorm2016.1\
```

Refer to the page Project and IDE Settings .

Linux and the other UNIX systems

Product directory starting with dot can be found in the user home directory. The pattern is:

```
~/.<PRODUCT><VERSION>
```

~ is an alias for the home directory, for example `/home/john`.

OS X

- Configuration

```
~/Library/Preferences/<PRODUCT><VERSION>
```

- Caches

```
~/Library/Caches/<PRODUCT><VERSION>
```

- Plugins

```
~/Library/Application Support/<PRODUCT><VERSION>
```

- Logs

```
~/Library/Logs/<PRODUCT><VERSION>
```

where <PRODUCT> is PhpStorm.

Managing Projects under Version Control

This section contains information related to sharing PhpStorm project files with the other developers:

- Directory based project format
- Sharing run/debug configurations
- Sharing inspection profiles
- Project settings files to share

Directory based project format

The project settings are stored in the `.idea` directory. This format is used by all PhpStorm versions by default. Here is what you need to share:

- All the files under `.idea` directory in the project root except `workspace.xml` and `tasks.xml`, storing user-specific settings.

Sharing run/debug configurations

You might want to share run/debug configurations . To do that, just select the check box Share in the selected run/debug configuration dialog box .

The shared run/debug configurations are kept in separate xml files under `.idea\runConfigurations` folder, while the local run/debug configurations are kept in the `.idea\workspace.xml` .

Sharing inspection profiles

To share inspection profiles, make sure to select the check box Share profile on the Inspections page of the Settings dialog.

The shared inspection profiles are stored in separate xml files under `.idea\inspectionProfiles` folder, while the local profiles are kept in the `.idea\workspace.xml` .

Project settings files to share

The `config` directory has several subfolders that contain xml files with your personal settings. You can easily share your preferred keymaps, color schemes, etc. by copying these files into the corresponding folders on another PhpStorm installation. Prior to copying, make sure that PhpStorm is not running, because it can erase the newly transferred files before shutting down.

The following is the list of some of the subfolders under the `config` folder, and the settings contained therein.

Folder name	User Settings
`codestyles`	Contains code style schemes .
`colors`	Contains editor colors and fonts customization schemes.
`filetypes`	Contains user-defined file types .
`inspection`	Contains code inspection profiles .
`keymaps`	Contains PhpStorm keyboard shortcuts customizations.
`options`	Contains various options, for example, feature usage statistics and macros.
`templates`	Contains user-defined live templates .
`tools`	Contains configuration files for the user-defined external tools .
`shelf`	Contains shelved changes .

Warning! Be careful about sharing the following:

- `dataSources.ids` , `datasources.xml` - these files can contain database passwords.

Also, consider not sharing the following:

- The user dictionaries folder to avoid conflicts if another developer has the same name.

Color-Deficiency Adjustment

In this section:

- Light editor schemes
- Darcula scheme
- Test runner adjustment

Light editor schemes

To people with red-green color deficiency, green, red and their hues might look the same. In the default color scheme, red is reserved for the errors and green for the strings:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

The simulation below shows how the same code fragment will look for a person with green color deficiency. Strings, annotations and unknown symbols are all the same color. The wavy error underline is lighter and less noticeable:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

The Adjust for color deficiency option changes colors so that they can be differentiated by a person with green or red color deficiency. Strings and annotations are shades of blue, and orange is reserved for the error states:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

Then, with the option Adjust for color deficiency turned on, and green color deficiency is selected, the simulation for the green color deficiency will look like the following:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

Highlight for read/write states of identifiers at caret are well distinguished for the persons without color deficiency:

```
Class callerClass = null;
for (final Object o : element.getChildren()) {
  Element listElement = (Element)o;
  if (ATTR_LIST.equals(listElement.getName())) {
    if (callerClass == null) {
      callerClass = ReflectionUtil.findCallerClass(2);
      assert callerClass != null;
    }
```

However, these states become indistinguishable for the persons with color deficiency:

```
Class callerClass = null;
for (final Object o : element.getChildren()) {
  Element listElement = (Element)o;
  if (ATTR_LIST.equals(listElement.getName())) {
    if (callerClass == null) {
      callerClass = ReflectionUtil.findCallerClass(2);
      assert callerClass != null;
    }
```

With the Adjust for color deficiency option, the difference between read and write states remains visible. In the example below you see the green color deficiency simulation:

```
Class callerClass = null;
for (final Object o : element.getChildren()) {
  Element listElement = (Element)o;
  if (ATTR_LIST.equals(listElement.getName())) {
    if (callerClass == null) {
      callerClass = ReflectionUtil.findCallerClass(2);
      assert callerClass != null;
    }
```

Folded text highlight in the default scheme is easily noticeable for the persons without color deficiency:

```
String ABSTRACTION_GROUP_NAME = InspectionsBundle.message("group.names.abstraction.issues");
String ASSIGNMENT_GROUP_NAME = "Assignment issues";
String BUGS_GROUP_NAME = "Probable bugs";
```

But this is a challenge for the persons with color deficiency - it is too light to notice:

```
String ABSTRACTION_GROUP_NAME = InspectionsBundle.message("group.names.abstraction.issues");
String ASSIGNMENT_GROUP_NAME = "Assignment issues";
String BUGS_GROUP_NAME = "Probable bugs";
```

With the Adjust for color deficiency option it becomes noticeable:

```
String ABSTRACTION_GROUP_NAME = InspectionsBundle.message("group.names.abstraction.issues");
String ASSIGNMENT_GROUP_NAME = "Assignment issues";
String BUGS_GROUP_NAME = "Probable bugs";
```

Darcula scheme

The non-adjusted Darcula text looks as follows:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

The difference between the various elements of text is not noticeable for the persons with color deficiency:

```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

Now turn on the option Adjust for color deficiency.

Compare the view of the editor for the persons without color deficiency:

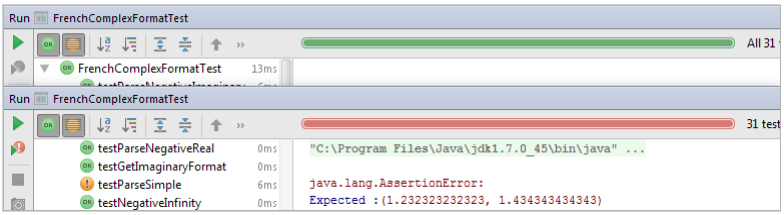
```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

and that for the persons with color deficiency:

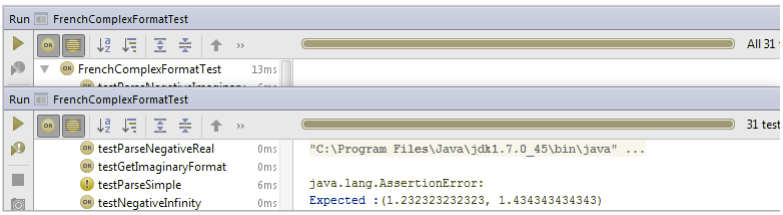
```
@NonNls private static final String PROJECT_DEFAULT_PROFILE_NAME = "Project Default";

public DefaultProjectProfileManager(@NotNull final Project project,
                                    @NotNull ApplicationProfileManager applicationProfileManager,
                                    @NotNull DependencyValidationManager holder)
  myProject = project;
  myHolder = holder;
  myApplicationProfileManager = applicationProfileManager;
}
```

Test runner adjustment

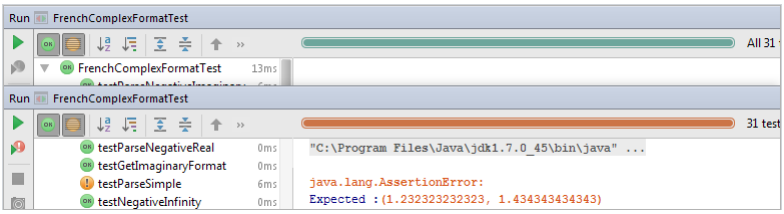
Colors of the test runner progress have also been adjusted. The usual progress bar color is indistinguishable for a person with green or red color deficiency. Compare the view of the test runner for the person without color deficiency:



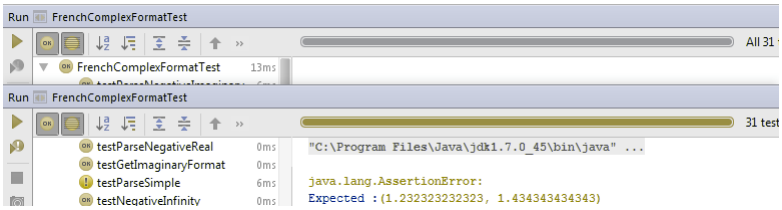
with that for the persons with color deficiency:



Turn on the option Adjust for color deficiency, and compare the view of the test runner for the person without color deficiency:



with that for a person with color deficiency:



Index of Menu Items

PhpStorm menu structure doesn't align with PhpStorm help structure. This page lists PhpStorm menu items, linked to the corresponding help topics.

> **Tip** This table corresponds to the Windows platform.
>
> Any additional plugins and external tools make changes to the main menu.

In this section:

- File
- Edit
- View
- Navigate
- Code
- Refactor
- Run
- Tools
- VCS
- Window
- Help

File

Menu item	Keyboard shortcut	Description
New...	Alt+Insert	Use this command to create a new element in a project . This command is only available in the corresponding context.
Open...		Use this command to open the specified directory, or an existing PhpStorm project. A directory that contains a project is marked with icon. Refer to the section Opening, Reopening, and Closing Projects . This command is duplicated with icon on the main toolbar.
Open URL...		Use this command to open source code of a web page in the editor.
Open Recent		Use this command to open one of the recent projects. Refer to the reopening projects procedure.
Close Project		Use this command to close the current project . Refer to the sections Opening Multiple Projects and Opening, Reopening and Closing Projects for details.
Settings... (on Windows/*NIX)/PhpStorm Preferences (on OS X)	Ctrl+Alt+S	Use this command to change the project and IDE configurations in the Settings/Preferences dialog . See also the section Configuring Project and IDE Settings . This command is duplicated with icon on the main toolbar. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu and has the name Preferences.
Default Settings...		Choose this command to change settings that will apply to all newly created projects. Refer to the section Accessing Default Settings .
Import Settings...		Choose this command to import settings from an archive .
Export Settings...		Choose this command to export settings to an archive .
Settings Repository...		Choose this command to invoke the Settings Repository dialog.
Save All	Ctrl+S	Choose this command to save all changes , when editing is over. This command is duplicated with icon on the main toolbar.
Synchronize	Ctrl+Alt+Y	Choose this command to check the PhpStorm caches and bring them up-to-date by keeping in sync with external changes. This command is duplicated with icon on the main toolbar.
Invalidate Caches/Restart...		Choose this command to clean the system cache .
Export to HTML...		Use this command results to save selected files in HTML format .
Print...		Choose this command to print selected file on the default printer. Refer to the Print dialog description.
Add to Favorites		Use this command to add the selected files to the list of Favorites . Click the right arrow to select the list of favorites you want to be modified. Refer to the description of the Favorites tool window .
File Encoding		Use this command to change encoding of an individual file . See also the section Encoding .
Line Separators		Use this command to select the desired line separator style. Refer to the section Configuring Line Separators .
Make File Read-Only / Make File Writable		Use these toggle commands to change read-only status of a file selected in the Project tool window, or currently active in the editor. If a file is made read-only, it is marked with , and doesn't allow editing. You can also toggle read-only attribute of a file in the Status bar .
Power Save Mode		Use this mode if you are working with a laptop. If Power-Save mode is on, then the background processes are turned off, to minimize the power consumption. You can also turn this mode on or off by clicking in the Status bar .
Exit		Choose this command to quit PhpStorm. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu and has the name Quit PhpStorm.

Edit

Menu item	Keyboard shortcut	Description
Undo <action>	Ctrl+Z	Use this command to roll actions back . This command is duplicated with icon on the main toolbar.
Redo <action>	Ctrl+Shift+Z	Use this command to repeat the last actions . This command is duplicated with icon on the main toolbar.
Cut	Ctrl+X	Choose this command to take the selected characters to the clipboard and delete them. Refer to the section Cutting, Copying, and Pasting . This command is duplicated with icon on the main toolbar.
Copy	Ctrl+C	Choose this command to take the selected characters to the clipboard. Refer to the section Cutting, Copying, and Pasting . This command is duplicated with icon on the main toolbar.
Copy Path	Ctrl+Shift+C	Choose this command to take the path to the selected symbol to the clipboard. Refer to the section Cutting, Copying, and Pasting .
Copy as Plain Text	Ctrl+Shift+C	Choose this command to take the selected fragment to the clipboard without formatting. Refer to the section Settings/Preferences | Editor | General .
Copy Reference	Ctrl+Shift+Alt+C	Choose this command to take a reference to a symbol to the clipboard. Refer to the section Cutting, Copying, and Pasting .
Paste	Ctrl+V	Choose this command to place the latest entry from the Clipboard at the insertion point. Refer to the section Cutting, Copying, and Pasting . This command is duplicated with icon on the main toolbar.
Paste from History...	Ctrl+Shift+V	Choose this command to place at the insertion point the selected entry from the Clipboard. Refer to the section Cutting, Copying, and Pasting .
Paste Simple	Ctrl+Shift+Alt+V	Choose this command to place the last entry from the Clipboard at the insertion point as plain text. Refer to the section Cutting, Copying, and Pasting .
Delete	Delete	Choose this command to delete the selected files, or folder from the project tool window, or selected fragment of text from the active editor.
Find		Point to this node to reveal the sub-menu of search commands:
Find/Replace	Ctrl+F / Ctrl+R	Find or replace text in a current file . These commands are duplicated by and icons on the main toolbar.
Find Next/Find Previous (Move to Next/Previous Occurrence)	F3 / Shift+F3	Use these commands to navigate through the search results in a file. See Finding and Replacing Text in File .
Find Word at Caret	Ctrl+F3	Use this command to jump to the next occurrence of the word where the caret rests. See Finding and Replacing Text in File .
Select All Occurrences	Ctrl+Shift+Alt+J	Use this command to find and select all the occurrences of an item.
Add Selection for Next Occurrence	Alt+J	Use this command to select the next occurrence of an item.
Unselect Occurrence	Shift+Alt+J	Use this command to remove selection from the last selected occurrence of an item.
Find in Path/Replace in Path	Ctrl+Shift+F / Ctrl+Shift+R	Use these commands to search for, and replace a text fragment in a whole project. Refer to the section Finding and Replacing text in Project .
Search/Replace Structurally		Use these commands to perform structural search or replace. Refer to the section Structural Search and Replace for details.
Find Usages	Alt+F7	Use this command to search for the usages of a symbol across an entire project. Refer to the section Finding Usages in Project .
Find Usages Settings	Ctrl+Shift+Alt+F7	Use this command to search for the usages of a symbol across an entire project, after setting the desired search options. Refer to the section Finding Usages in Project .
Show Usages	Ctrl+Alt+F7	Use this command to bring up a list of the usages of a symbol across the whole project. Refer to the section Viewing Usages of a Symbol .
Find Usages in a File	Ctrl+F7	Refer to the section Finding Usages in the Current File .
Highlight Usages in a File	Ctrl+Shift+F7	Use this command to visualize usage of a symbol in the current file. Refer to the section Highlighting Usages .
Recent Find Usages	Ctrl+E	Choose this command to view the recent search results. Refer to Viewing Recent Find Usages .
Find by XPath...	Ctrl+Alt+X, F	Use this command to find occurrences of certain XPath expressions in all XML files in a specific scope. Refer to XPath and XSLT Support .
Macros		Point to this node to reveal the sub-menu of the macros-related commands. Refer to the section Using Macros in the Editor .
Column Selection Mode	Shift+Alt+Insert	Use this command to toggle between column selection and line selection modes. Refer to the section Selecting Text in the Editor .
Select All	Ctrl+A	Choose this command to select all contents of the current file. Refer to the section Selecting Text in the Editor .
Extend Selection	Ctrl+W	Choose this command to select the current word. Use this command successively to extend selection. Refer to the section Selecting Text in the Editor .
Shrink Selection	Ctrl+Shift+W	Choose this command to unselect the currently selected word. Use this command successively to shrink selection. Refer to the

		section Selecting Text in the Editor .
Join Lines	Ctrl+Shift+J	Choose this command to join lines or literals .
Fill Paragraph		Choose this command to create soft wraps in a paragraph.
Duplicate Lines	Ctrl+D	Choose this command to duplicate a line or fragment of text. Refer to Adding, Deleting and Moving Code Elements .
Indent Selection/Unindent Selection	Tab / Shift+Tab	Choose this command to change indentation of the line at caret. Refer to the section Changing Indentation .
Toggle Case	Ctrl+Shift+U	Choose this command to change case of the selection. See Toggling Case .
Convert Indents		Point to this node to reveal the sub-menu of the possible indentation and toggle indentation style. Refer to Changing Indentation .
Encode XML/HTML Special Characters		Choose this command to convert the selected special character to its HTML name in the format `&char;` .
Edit as Table		Choose this command to invoke the table editor for the current documents.

View

Menu item	Keyboard shortcut	Description
Tool Windows		Point to this node to reveal the list of the available tool windows. Refer to the section Manipulating the Tool Windows .
Quick Definition	Ctrl+Shift+I	Choose this command to open the quick definition pop-up. Refer to the section Viewing Definition .
Quick Documentation	Ctrl+Q	Choose this command to view quick documentation pop-up window.
Parameter Info	Ctrl+P	Choose this command to view method parameter information .
Context Info	Alt+Q	Choose this command to show the current cursor position , if it runs out of the visible editor pane.
Jump to Source	F4	Choose this command to edit a file selected in a tool window. The file opens in the editor.
Recent Files	Ctrl+E	Choose this command to show the pop-up list of recently opened files and tool windows, and navigate to them.
Recently Changed Files	Ctrl+Shift+E	Choose this command to show the pop-up list of recently changed files and navigate to them .
Recent Changes	Shift+Alt+C	Choose this command to open the pop-up list of recent changes .
Compare with Clipboard		Choose this command to compare the file currently opened in the editor with the contents of the system clipboard. See Comparing Files .
Quick Switch Scheme	Ctrl+Back Quote	Choose this command to switch between schemes .
Toolbar		Select or clear this check command to show or hide the main toolbar .
Tool Buttons		Select or clear this check command to show or hide the tool window buttons .
Status Bar		Select or clear this check command to show or hide the Status toolbar .
Navigation Bar		Select or clear this check command to show or hide the Navigation bar .
Enter/Exit Presentation Mode		Choose this command to enter or exit presentation mode .
Enter/Exit Distraction Free Mode		Choose this command to enter or exit distraction-free mode .
Enter/Exit Full Screen		Choose this command to enter or exit full screen mode .

Navigate

Menu item	Keyboard shortcut	Description
Class/File/Symbol	Ctrl+N / Ctrl+Shift+N Ctrl+Shift+Alt+N	Choose these commands to find and jump to a class, file, or symbol by name .
Custom Folding...	Ctrl+Alt+Period	Choose this command to navigate between custom regions .
Line...	Ctrl+G	Choose this command to navigate to the specified line of code .
Back/Forward	Ctrl+Alt+Left / Ctrl+Alt+Right	Choose these commands to go through the history of the recently navigated items . These commands are duplicated with and buttons on the main toolbar.
Last/Next Edit Location	Ctrl+Shift+Backspace	Choose these commands to jump to the latest edit location and back.
Bookmarks		Point to this node to reveal the sub-menu of commands related to using bookmarks .
Select In...	Alt+F1	Choose this command to select the desired component from the pop-up list of possible targets .
Jump to Navigation Bar	Alt+Home	Choose this command to navigate across your project using the Navigation Bar .
Declaration	Ctrl+B	Choose this command to jump to a declaration of a symbol .
Implementation(s)	Ctrl+Alt+B	Choose this command to jump to an implementation of a method .
Type Declaration	Ctrl+Shift+B	Choose this command to jump to the type declaration of a symbol .
Super Method	Ctrl+U	Choose this command to jump to a super method of the method at caret.
Test	Ctrl+Shift+T	Choose this command to navigate to an existing test, or create a test.
Related Symbol...	Ctrl+Alt+Home	.
File Structure	Ctrl+F12	Choose this command to navigate through the source code using the File Structure view .
File Path	Ctrl+Alt+F12	See Navigating to File Path .

Type/Method/Call Hierarchy	Ctrl+H / Ctrl+Shift+H / Ctrl+Alt+H	Choose these commands to navigate using the hierarchy views. Refer to the sections Viewing Structure and Hierarchy of the Source Code .
Next/Previous Highlighted Error	F2 Shift+F2	Choose these commands to navigate through the highlighted errors .
Next/Previous Change	Ctrl+Shift+Alt+Down Ctrl+Shift+Alt+Up	Choose these commands to navigate though the change markers (when VCS integration is enabled).
Next/Previous Method	Alt+Down Alt+Up	Choose these commands to go up and down through the methods and tags .

Code

Menu item	Keyboard shortcut	Description
Override Methods...	Ctrl+O	Choose this command to override a method.
Implement Methods...	Ctrl+I	Choose this command to implement a method.
Generate...	Alt+Insert	Choose this command to create a new element. See (depending on the context) Populating Projects , Generating Code .
Surround With...	Ctrl+Alt+T	Choose this command to surround a logical fragment with code construct .
Unwrap/Remove...	Ctrl+Shift+Delete	Choose this command to unwrap an expression from enclosing statements .
Completion		Point to this node to reveal the nested auto-completion commands.
Folding		Point to this node to reveal the nested folding commands.
Insert Live Template...	Ctrl+J	Choose this command to create code constructs by live templates .
Surround with Live Template...	Ctrl+Alt+J	Choose this command to create code constructs using surround templates .
Comment with Line Comment	Ctrl+Slash	Choose this command to comment an entire line of code. See Commenting and Uncommenting Blocks of Code .
Comment with Block Comment	Ctrl+Shift+Slash	Choose this command to comment out a block of code. See Commenting and Uncommenting Blocks of Code .
Reformat Code...	Ctrl+Alt+L	Choose this command to perform code reformatting. See Reformatting Source Code .
Auto-Indent Lines	Ctrl+Alt+I	Choose this command to change indentation .
Optimize Imports...	Ctrl+Alt+O	Choose this command to optimize import statements. See Optimizing Imports .
Rearrange Code		Choose this command to rearrange code according to the arrangement rules. See Rearranging Code Using Arrangement Rules .
Move Statement Up/Down	Ctrl+Shift+Up / Ctrl+Shift+Down	Choose this command to move a statement up or down .
Move Element Left/Right	Ctrl+Shift+Alt+Left / Ctrl+Shift+Alt+Right	Choose this command to move element at caret left or right.
Move Line Up/Down	Shift+Alt+Up / Shift+Alt+Down	Choose this command to move a line at caret up or down .
Inspect Code...		Choose this command to run an inspection .
Code Cleanup		Choose this command to open the dialog Specify Code Cleanup Scope Dialog .
Run Inspection by Name...	Ctrl+Shift+Alt+I	Choose this command to run the specified inspection .
Configure Current File Analysis...	Ctrl+Shift+Alt+H	Choose this command to change highlighting level of the current file .
View Offline Inspection Results...		Choose this command to see inspection results stored on your computer.
Locate Duplicates...		Choose this command to find code duplicates. Refer to Analyzing Duplicates .

Refactor

Note that the composition of this menu item depends upon the current context.

Menu item	Keyboard shortcut	Description
Refactor This...	Ctrl+Shift+Alt+T	Choose this command to open a pop-up menu of the refactorings available in the current context. Refer to the section Refactoring Source Code .
Rename...	Shift+F6	Choose this command to rename an element .
Change Signature...	Ctrl+F6	Choose this command to perform change signature refactoring .
Move...	F6	Choose this command to move a symbol to the specified location.
Copy...	Ctrl+C	Choose this command to create a copy of an element in the specified location.
Safe Delete	Alt+Delete	Choose this command to delete a symbol , performing search for its usages.
Extract		Choose this command to perform one of the extract refactorings.
Inline...	Ctrl+Alt+N	Choose this command to perform inline refactoring .
Pull Members Up...		Choose this command to perform pull members up refactoring .
Push Members Down		Choose this command to perform push members down refactoring .

Run

Menu item	Keyboard shortcut	Description
Run <current run/debug configuration>	Shift+F10	Choose this command to run the current with the corresponding temporary run/debug configuration. This command is duplicated with icon on the main toolbar.
Debug <current run/debug configuration>	Shift+F9	Choose this command to debug the current with the corresponding temporary run/debug configuration. This command is duplicated with icon on the main toolbar.
Run <current run/debug configuration> with Coverage		Choose this command to run with coverage the current with the corresponding temporary run/debug configuration. This command is duplicated with icon on the main toolbar.
Run...	Shift+Alt+F10	Choose this command to select the desired run/debug configuration, and then launch it. Refer to the section Creating and Editing Run/Debug Configurations .
Debug...	Shift+Alt+F9	Choose this command to select the desired run/debug configuration, and then launch it in debugging mode. Refer to the section Creating and Editing Run/Debug Configurations .
Edit Configurations...		Choose this command to change run/debug configurations .
Import Test Results		Choose this command to import test results from a file.
Stop	Ctrl+F2	Choose this command to terminate execution of a run/debug configuration . This command is duplicated with icon in the toolboxes of the Run and Debug tool windows.
Show Running List		Choose this command to display a popup that lists all currently running/debugging applications. Refer to the section Viewing Running Processes .
Stepping Commands		These commands become enabled with the debugger session on. Refer to the section Stepping Through the Program . See also descriptions of the stepping toolbar buttons in the Debug tool window reference.
Pause Program		Choose this command to pause output of the current run or debug session. This command is duplicated with icon in the toolboxes of the Run and Debug tool windows. Note that the button is not available for Run/Debug Configuration: Node JS , Run/Debug Configuration: Node JS Remote Debug , and Run/Debug Configuration: NodeUnit .
Resume Program	F9	Choose this command to resume the debugger session with the selected run/debug configuration. This command is duplicated with icon in the toolbox of the Debug tool windows.
Evaluate Expression	Alt+F8	Choose this command to evaluate expression during the debug session.
Quick Evaluate Expression	Ctrl+Alt+F8	Choose this command to perform quick evaluation of an expression in the editor during the debug session.
Show Execution Point	Alt+F10	Choose this command to show execution point during the debug session.
Toggle Line Breakpoint	Ctrl+F8	Choose this command to turn on or off a line breakpoint. Refer to the section Creating Line Breakpoints .
Toggle Temporary Line Breakpoint	Ctrl+Shift+Alt+F8	Choose this command to turn on or off a temporary line breakpoint. Refer to the section Creating Line Breakpoints .
View Breakpoints...	Ctrl+Shift+F8	Choose this command to show all available breakpoints and change them in the Breakpoints dialog .

Tools

Note that composition of the menu Tools depends on the enabled plugins and external tools .

Menu item	Keyboard shortcut	Description
Tasks and Contexts		Point to this node to reveal the sub-menu of commands related to tasks and contexts management . See also Tasks reference page.
Save File as Template		Choose this command to save the current file as a template file .
New Scratch File	Ctrl+Shift+Alt+Insert	Choose this command to create a new scratch file .
IDE Scripting Console		Choose this command to launch the interactive scripting console.
Analyze Stacktrace...		Choose this command to analyze external stacktrace .
Capture Memory Snapshot		Choose this command to get the memory state of the profiled application.
Show Code Coverage Data	Ctrl+Alt+F6	Choose this command to view coverage results . Refer to the section Code Coverage .
Vim Emulator		Select this check command to enable or disable Vim emulation. This command only appears when Vim plugin is installed and enabled.
Reconfigure Vim Keymap		This command is only visible, when Vim Emulator is checked. Choose it to select a different base keymap for the Vim emulator.
Deployment		Point to this node to reveal the sub-menu of deployment-related commands. Refer to the section Working with Web Servers: Copying Files .
Open terminal		Choose this command to run the embedded local terminal .
Start SSH Session		Choose this command to launch a terminal on a remote SSH server. Refer to the section Running SSH Terminal .
Test RESTful Web Service		Choose this command to compose and run requests to a RESTful web service. Refer to the section Testing RESTful Web Services .
Vagrant		Point to this node to reveal the sub-menu of standard Vagrant actions. See Vagrant: Working with Reproducible Development Environments .

VCS

Note that the VCS menu contains different commands, depending on the enabled version control system. The following table shows the menu commands available when no version control integration is enabled.

Menu item	Keyboard shortcut	Description
Local History		Point to this node to reveal the list of commands related to working with the Local History .
Enable Version Control Integration...		Choose this command to associate a project root with one of the supported version control systems.
VCS Operations Popup	`Alt+Back Quote`	Choose this command to invoke the pop-up list of the most popular VCS actions .
Apply Patch...		Choose this command to apply a patch .
Checkout from Version Control		Point to this node to reveal the sub-menu of the checkout commands, specific for the supported version control systems. With no version control integration enabled, it is possible to check out from SVN , Mercurial , Git , GitHub , and CVS .
Import into Version Control		Point to this node to reveal the sub-menu of the import commands, specific for the supported version control systems. With no version control integration enabled, it is possible to import to SVN , Mercurial , Git , GitHub , and CVS .
Browse VCS Repository		Point to this node to reveal the sub-menu of the browse commands, specific for the supported version control systems. With no version control integration enabled, it is possible to browse Subversion, Git, and CVS repositories that are not associated with the currently opened project. – Browse CVS Repository: when you choose this option PhpStorm opens the Select CVS Root Configuration dialog, where you can select the relevant CVS root, see Configuring CVS Roots and Browsing CVS Repository . – Browse Git Repository Log: choose this option to view the log for a local Git repository that is associated with another project. When you select the relevant repository in the Select Path dialog box, PhpStorm adds a new Log tab to the Version Control tool window and shows the log for the selected repository. The name of the project associated with the selected repository is displayed in the tab title, when you hover the mouse over this tab, the full path to the repository is shown in a tooltip. – Browse Subversion Repository: when you choose this option PhpStorm opens the SVN Repositories tool window where you can view, add, and edit location of SVN repositories, see Browsing Subversion Repository and Browsing Contents of the Repository .

It is important to note that with VCS integration enabled, the composition of the VCS menu is different. Refer to the following help sections for details:

- Version Control concepts
- Common VCS procedures
- VCS-specific procedures
- Version Control reference

Window

Menu item	Keyboard shortcut	Description
Store Current Layout as Default		Choose this command to save the current way the tool windows are arranged .
Restore Default Layout	`Shift+F12`	Choose this command to restore the initial way the tool windows are arranged .
Active Tool Window		Choose this command to reveal the sub-menu of commands, related to the active tool window . Refer to the sections PhpStorm Tool Windows , Manipulating the Tool Windows , Specifying the Appearance Settings for Tool Windows , Viewing Modes .
Editor Tabs		Choose this command to reveal the sub-menu of commands, related to the editor tabs. Refer to the section Managing Editor Tabs . Note that these commands can be also found on the context menu of an editor tab.
Background Tasks		Choose this command to reveal the sub-menu of commands, related to performing tasks in background. Refer to the section Working with Background Tasks .
Next Project Window / Previous Project Window	`Ctrl+Alt+Close Bracket` `Ctrl+Alt+Open Bracket`	Choose this command to switch between currently opened projects.
<project>		Select project to be shown in the active window.

Help

Menu item	Keyboard shortcut	Description
Find Action	`Ctrl+Shift+A`	Choose this command to invoke an action by its name .
Keymap Reference		Choose this command to see the PhpStorm shortcuts map in PDF format.
Demos and Screencasts		Choose this command to see the PhpStorm demo videos and screencasts on YouTube .
Help		Choose this command to visit PhpStorm online Help topics.
Tutorials		Choose this command to open the page of tutorials .
Tip of the Day		Choose this command to show an arbitrary tip. Refer to the section Using Tips of the Day .
Productivity Guide		Choose this command to show productivity guide .
Submit Feedback		Choose this command to report your overall impression of PhpStorm to the support service. Refer to the section Reporting Issues and Sharing Your Feedback .
Show Log in Explorer/Finder		Choose this command to find PhpStorm's log. Refer to the section Reporting Issues and Sharing Your Feedback for details.
Edit Custom Properties		Choose this command to open the custom file `idea.properties` , located under the user home. If this file does not exist, PhpStorm suggests to create it. Refer to the section Tuning PhpStorm for details.

Edit Custom VM Options	Choose this command to open the custom file `*.vmoptions`, located under the user home. If this file does not exist, PhpStorm suggests to create it. Refer to the section Tuning PhpStorm for details.
Debug Log Settings	Choose this command to change logging level for a category. Choosing this command leads to opening the Custom Debug Log Configuration dialog box, where you have to type the log categories names, separated with new lines. Refer to the section Reporting Issues and Sharing Your Feedback .
Diagnostics	This command and the items below it are available only when the Performance plugin is installed and activated. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins . Use the commands under this menu item to automatically capture snapshots and collect other data related to performance and send this data to JetBrains Support . When you click Diagnostics, a drop-down menu with the following items is displayed: – Enable CPU Profiler: choose this option to activate the CPU profiler and capture CPU snapshots. The CPU profiles remains active and the option not available till you restart PhpStorm. – Change Memory Settings: choose this option to edit the `-Xmx` setting and thus increase the heap size (the amount of available memory), see Tuning PhpStorm for details. – Execute Performance Script: choose this option to run special predefined scenarios with the CPU profiler enabled. After a scenario is executed it generates a CPU snapshot that can be used by the support team to analyze your performance issue. – Profile Indexing: when you choose this option, CPU profiler is active and captures a CPU snapshot from the moment you reopen the current project till the moment indexing is completed. The snapshot can be used by the support team to analyze a performance issue that arize during indexing. – Show Performance Summary: choose this option to collect most relevant information that can affect performance and at the same time does not reveal any private info about the current project and the current file. Learn more at Performance testing plugin in PhpStorm
Settings Summary	Choose this command to collect the PhpStorm settings and show all sorted by category so it is easy to locate the settings that are related with a specific problem (debugger, or deployment, or something else does not work) and send this summary to JetBrains Support . To open a request form, click Open Support Request in the dialog box that opens. This command is available only when the Performance plugin is installed and activated. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
Register...	Choose this command to register PhpStorm.
Check for Updates...	Choose this command to obtain information about the current version, and the availability of newer versions of PhpStorm. Refer to Updates page. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu.
About	Choose this command to obtain information about the current version of PhpStorm, current build, etc. Press Escape to close the pop-up window. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu.

Tuning PhpStorm

In this part:

- Tuning PhpStorm
  - Changing PhpStorm properties
  - Managing *.vmoptions file
    - Example: Increasing the heap size
  - Managing idea.properties file
    - Example: Changing case of unicode literals
  - Specifying custom JDK, properties or vmoptions files across platforms
- File IDEA.Properties'

Changing PhpStorm properties

You can make PhpStorm work faster by editing the default Java Virtual Machine (JVM) settings. These operations require certain understanding of the JVM functionality and may cause unexpected problems. Therefore it is strongly recommended that you address the JetBrains Support whereupon you will be provided with detailed instructions regarding which settings you need to update and what values should be assigned to them.

See also Reporting Issues and Sharing Your Feedback .

PhpStorm makes it possible to change `*.vmoptions` and `idea.properties` files without editing them in the PhpStorm installation folder.

To create an empty `idea.properties` file or to copy `*.vmoptions` file, choose the Help | Edit Custom Properties or Help | Edit Custom VM Options on the main menu respectively. Refer to the menu items description for details.

Managing *.vmoptions file

The location of the `*.vmoptions` file depends on the operating system you are currently using:

- For Windows systems: `<PhpStorm installation folder>/bin/phpstorm.vmoptions` or `<PhpStorm installation folder>/bin/phpstorm64.vmoptions`
- For *NIX systems: `<PhpStorm installation folder>/bin/phpstorm.vmoptions` or `<PhpStorm installation folder>/bin/phpstorm64.vmoptions`
- For OS X systems, you have to make a copy of the `phpstorm.vmoptions` file in the IDE preferences folder and then edit this copy. The reason is that app bundle is signed and you should not modify any files inside the bundle.
  - Since version 2016.1, the file `/Applications/PhpStorm.app/Contents/bin/phpstorm.vmoptions` should be copied to `~/Library/Preferences/PhpStorm/phpstorm.vmoptions` .
  - For earlier versions, the file `/Applications/PhpStorm<version>.app/Contents/bin/idea.vmoptions` should be copied to `~/Library/Preferences/PhpStorm<version>/idea.vmoptions` .

Tip Mind the abbreviation scheme of the PhpStorm name: `xx` corresponds to the PhpStorm version.

To avoid editing files in the PhpStorm installation folder, one should:

1. Do one of the following:
   - Use the main menu command Help | Edit Custom VM Options to create a copy of the `phpstorm.vmoptions` file in the user home. Refer to Edit Custom VM Options for details.
   - Copy the existing file from the PhpStorm installation folder somewhere and save the path to this location in the environment variable `WEBIDE_VM_OPTIONS` .
   - Copy the existing file `<PhpStorm installation folder>/bin/phpstorm$BITS.vmoptions` from the PhpStorm installation folder into the location under user home.
2. Edit this file in the new location.

If `WEBIDE_VM_OPTIONS` environment variable is defined, or the `*.vmoptions` file exists, then this file is used instead of the one located in the PhpStorm installation folder.

Example: Increasing the heap size

To increase PhpStorm heap size, you should copy the original `phpstorm.vmoptions` file to the above-mentioned location, and then modify the `-Xmx` setting.

Managing IDEA.properties file

The file `idea.properties` , located in the `bin` directory of the PhpStorm installation folder, should not be edited. Instead of editing the original `idea.properties` , create file `idea.properties` in the location specified below, open it for editing and add the required properties.

So, depending on your platform:

- For Windows: in `%USERPROFILE%\.PhpStormXX`
- For *NIX: in `~/.PhpStormXX`
- For OS X: in `~/Library/Preferences/PhpStormXX`

Example: Changing case of unicode literals

PhpStorm allows defining whether non-ascii characters should use literals like `'\u00AB'` or `'\00ab'` .

This behavior is controlled by the system property `idea.native2ascii.lowercase` . By default, upper case characters are used.

If it is desirable to use lower case characters, create the file `idea.properties` in the location specified above, open it for editing and add the following line:

```
idea.native2ascii.lowercase=true
```

Specifying custom JDK, properties or vmoptions files across platforms

A custom JDK, `*.properties` and `*.vmoptions` files are specified across platforms in a unified way.

All the launchers look at the following environment variables:

- `$<IDE-NAME>_JDK` (`<IDE-NAME>_JDK_64`)
- `$<IDE-NAME>_PROPERTIES`
- `$<IDE-NAME>_VM_OPTIONS`

File 'idea.properties'

The file `idea.properties`, located in the `bin` directory of the PhpStorm installation folder, should not be edited. Instead of editing the original `idea.properties`, create file `idea.properties` in the location specified below, open it for editing and add the required properties.

So, depending on your platform:

- For Windows: in `%USERPROFILE%\.PhpStorm XX`
- For *NIX: in `~/.PhpStorm XX`
- For OS X: in `~/Library/Preferences/PhpStorm XX`

You can create an empty file `idea.properties` and open it in the editor by choosing the Help | Edit Custom Properties command on the main menu.

Refer to the description of Edit custom properties command.

Name	Description	Property setting

Note The Windows users should use forward slashes. i.e. `c:/idea/system`.

${idea.home} macro	Use ${idea.home} macro to specify location relative to IDE installation home. Also use `${xxx}` where `xxx` is any java property (including defined in the previous lines of this file) to refer to its value.	
Path to the IDE config folder	Uncomment this option if you want to customize path to IDE config folder.	`idea.config.path=${user.home}/.PhpStorm/config`
Path to IDE system folder	Uncomment this option if you want to customize path to IDE system folder.	`idea.system.path=${user.home}/.PhpStorm/system`
Path to user installed plugins	Uncomment this option if you want to customize path to user installed plugins folder.	`idea.plugins.path=${user.home}/.PhpStorm/config/plugins`
Path to IDE logs folder	Uncomment this option if you want to customize path to IDE logs folder.	`idea.log.path=${user.home}/.PhpStorm/system/log`
Maximum file size	Maximum file size (kilobytes) IDE should provide code assistance for. The larger file is, the slower its editor works and higher overall system memory requirements are, if code assistance is enabled. Remove this property or set to a very large number, if you need code assistance for any files to be available, regardless of their size.	`idea.max.intellisense.filesize=2500`
Console cyclic buffer	This option controls console cyclic buffer: keeps the console output size not higher than the specified buffer size (Kb). Older lines are deleted. In order to disable cycle buffer, use `idea.cycle.buffer.size=disabled`	`idea.cycle.buffer.size=1024`
Launcher	Configure if a special launcher should be used when running processes from within IDE. Using Launcher enables "soft exit" and "thread dump" features.	`idea.no.launcher=false`
Classpath	To avoid too long classpath	`idea.dynamic.classpath=false`
ProcessCanceledException	Uncomment this property to prevent IDE from throwing ProcessCanceledException when user activity detected. This option is only useful for plugin developers, while debugging PSI related activities performed in background error analysis thread. DO NOT UNCOMMENT THIS UNLESS YOU'RE DEBUGGING THE IDE ITSELF. Significant slowdowns and lockups will happen otherwise.	`idea.ProcessCanceledException=disabled`
Pop-up window weight	There are two possible values of	

	idea.popup.weight property: "heavy" and "medium". If you have WM configured as "Focus follows mouse with Auto Raise" then you have to set this property to "medium". It prevents problems with the pop-up menus on some configurations.	`idea.popup.weight=heavy`
System anti-aliasing	Use default anti-aliasing in system, i.e. override value of "Settings|Editor|Appearance|Use anti-aliased font" option. May be useful when using Windows Remote Desktop Connection for instance.	`idea.use.default.antialiasing.in.editor=false`
Repaint	Disabling this property may lead to visual glitches like blinking and fail to repaint on certain display adapter cards.	`sun.java2d.noddraw=true`
Editor performance	Removing this property may lead to editor performance degradation under Windows.	`sun.java2d.d3d=false`
Slow scrolling	Workaround for slow scrolling in JDK6.	`swing.bufferPerWindow=false`
Editor performance under X Window	Removing this property may lead to editor performance degradation under X Window.	`sun.java2d.pmoffscreen=false`
Avoid long hangs	Workaround to avoid long hangs while accessing clipboard under OS X.	`ide.mac.useNativeClipboard=True`
Maximum load size	Maximum size (kilobytes) IDEA will load for showing past file contents - in Show Diff or when calculating Digest Diff	`idea.max.vcs.loaded.size.kb=20480`
Copy library jars	IDE copies library jars to prevent their locking. If copying is not desirable, specify "true"	`idea.jars.nocopy=false`
Start the JVM in debug mode	The VM option value to be used to start the JVM in debug mode. Some JREs define it in a different way (-XXdebug in Oracle VM)	`idea.xdebug.key=-Xdebug`
Switch into JMX 1.0 compatibility mode.	Uncomment this option to be able to run PhpStorm using J2SDK 1.5+ while working with application servers (like WebLogic) running 1.4.	`jmx.serial.form=1.0`
Fatal errors notifications	Change to 'enabled' if you want to receive instant visual notifications about fatal errors that happen to an IDE or plugins installed.	`idea.fatal.error.notification=disabled`

Getting Help

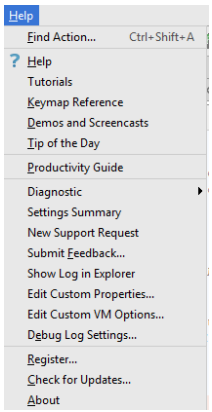
There are a lot of ways to obtain information about PhpStorm and its features, report issues and get support.

In this section:

- Getting Help
  - Overview
  - Menu commands
- Using Help Topics
- Using Tips of the Day
- Using Online Resources
- Using Productivity Guide
- Reporting Issues and Sharing Your Feedback
- Keymap Reference

Overview

Use the Help menu commands:



Menu commands

Menu item	Keyboard shortcut	Description
Find Action	`Ctrl+Shift+A`	Choose this command to invoke an action by its name .
Keymap Reference		Choose this command to see the PhpStorm shortcuts map in PDF format.
Demos and Screencasts		Choose this command to see the PhpStorm demo videos and screencasts on YouTube .
Help		Choose this command to visit PhpStorm online Help topics.
Tutorials		Choose this command to open the page of tutorials .
Tip of the Day		Choose this command to show an arbitrary tip. Refer to the section Using Tips of the Day .
Productivity Guide		Choose this command to show productivity guide .
Submit Feedback		Choose this command to report your overall impression of PhpStorm to the support service. Refer to the section Reporting Issues and Sharing Your Feedback .
Show Log in Explorer/Finder		Choose this command to find PhpStorm's log. Refer to the section Reporting Issues and Sharing Your Feedback for details.
Edit Custom Properties		Choose this command to open the custom file `idea.properties` , located under the user home. If this file does not exist, PhpStorm suggests to create it. Refer to the section Tuning PhpStorm for details.
Edit Custom VM Options		Choose this command to open the custom file `*.vmoptions` , located under the user home. If this file does not exist, PhpStorm suggests to create it. Refer to the section Tuning PhpStorm for details.
Debug Log Settings		Choose this command to change logging level for a category. Choosing this command leads to opening the Custom Debug Log Configuration dialog box, where you have to type the log categories names, separated with new lines. Refer to the section Reporting Issues and Sharing Your Feedback .
Diagnostics		This command and the items below it are available only when the Performance plugin is installed and activated. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins . Use the commands under this menu item to automatically capture snapshots and collect other data related to performance and send this data to JetBrains Support . When you click Diagnostics, a drop-down menu with the following items is displayed: – Enable CPU Profiler: choose this option to activate the CPU profiler and capture CPU snapshots. The CPU profiles remains active and the option not available till you restart PhpStorm. – Change Memory Settings: choose this option to edit the `-Xmx` setting and thus increase the heap size (the amount of available memory), see Tuning PhpStorm for details.

	– Execute Performance Script: choose this option to run special predefined scenarios with the CPU profiler enabled. After a scenario is executed it generates a CPU snapshot that can be used by the support team to analyze your performance issue. – Profile Indexing: when you choose this option, CPU profiler is active and captures a CPU snapshot from the moment you reopen the current project till the moment indexing is completed. The snapshot can be used by the support team to analyze a performance issue that arize during indexing. – Show Performance Summary: choose this option to collect most relevant information that can affect performance and at the same time does not reveal any private info about the current project and the current file. Learn more at Performance testing plugin in PhpStorm
Settings Summary	Choose this command to collect the PhpStorm settings and show all sorted by category so it is easy to locate the settings that are related with a specific problem (debugger, or deployment, or something else does not work) and send this summary to JetBrains Support . To open a request form, click Open Support Request in the dialog box that opens. This command is available only when the Performance plugin is installed and activated. The plugin is bundled with PhpStorm and activated by default. If the plugin is not activated, enable it on the Plugins page of the Settings / Preferences Dialog as described in Enabling and Disabling Plugins .
Register...	Choose this command to register PhpStorm.
Check for Updates...	Choose this command to obtain information about the current version, and the availability of newer versions of PhpStorm. Refer to Updates page. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu.
About	Choose this command to obtain information about the current version of PhpStorm, current build, etc. Press Escape to close the pop-up window. This command in available on Windows/Linux. On Mac OS it appears on the PhpStorm menu.

Using Help Topics

Documentation structure

PhpStorm documentation has the following structure:

Meet PhpStorm

This part contains system requirements and installation information, quick start guide that helps you get a grip of PhpStorm, the section First Steps with the tutorials that help you perform the most basic actions - create and run your first application.

How to

This contains information related to the platform features (such as, for example, using the PhpStorm editor , tool windows , or version control), and language- and framework-specific guidelines .

Reference

This part contains the miscellaneous information, which includes the basic concepts, or essentials , dialogs reference , icons reference , and more.

Online documentation

On the PhpStorm site, find online documentation:

PhpStorm web help

The online help makes it possible to find entries in the table of contents, browse documentation with the table of contents, rate topics and express your opinion. The layout of online documentation consists of:

Table of contents pane

This pane shows the table of contents.

- Use this pane to browse through the topics.
- Click button to show or hide this pane.
- If for some reason your browser fails to show actual table of contents, refresh the page.

Topics pane

This pane shows the topic that is currently selected in the table of contents.

- In the Keymap drop-down list, choose the platform you want to view keyboard shortcuts in (Windows/Linux, OS X etc.)

Finding a piece of information in the table of contents

1. Switch to the Table of contents pane.
2. In the Search PhpStorm help field, type your query.
3. Press Enter.
   The table contents shrinks to show the search results. Click the desired entry to show the corresponding page in the Topic pane.

Using Tips of the Day

Tips of the Day provide a collection of useful and interesting hints. They show up every time you start PhpStorm.

To show Tips of the Day

- Choose Help | Tip of the Day on the main menu.

To navigate through the collection of tips

- Use the Previous and Next buttons.

To suppress Tips of the Day

- In the Tips of the Day window, clear the check box Show Tips on Startup .

Using Online Resources

If the online PhpStorm documentation fails to answer your questions, you can find more information on the web. The following resources are available:

- Official JetBrains home page
- PhpStorm community
- PhpStorm Blog
- PhpStorm Twitter `@PhpStorm`
- PhpStorm Plugin Repository

Finally, do not miss the JetBrains TV (Help | JetBrains TV). On the JetBrains TV page, choose the PhpStorm channel, and watch screencasts.

Using Productivity Guide

PhpStorm smartly analyzes the features you use most often during your development sessions, and reminds you of the features you might have missed.

The Productivity Guide dialog, available in Help | Productivity Guide, displays the list of features with usage statistics and tips.

Besides analysing your personal usage of features, you can discover similar features that you've never used. One of the ways to do so is to sort features by Group, and look for unused features that are next to the frequently used ones. You can quickly check how to use the feature by selecting it and studying the corresponding tip that opens.

Reporting Issues and Sharing Your Feedback

PhpStorm provides various means to report problems and seek for assistance. In this section:

- Locating PhpStorm log
- Configuring PhpStorm log settings
- Reporting issues
- Sharing feedback
- Seeking assistance

Locating PhpStorm log

On certain occasions, you will be required to attach the PhpStorm log to an email to the support service. You can easily locate the log file as described below.

- On the main menu, choose Help | Show Log in ....

Configuring PhpStorm log settings

To avoid editing the `log.xml` file itself, PhpStorm suggests a handy dialog box to change logging level for a category. This file resides under the `bin` directory of PhpStorm installation.

1. On the main menu, choose Help | Configure Debug Log Settings.
2. In the dialog box that opens, type the log categories names, separated with new lines.

While editing `log.xml`, keep in mind the following:

- It is not recommended to change `log.xml`, because from time to time it causes problems with patches.
- Editing `log.xml` should be done in tight contact with the support service. The reason is that the users might be unaware of the modules names to be specified, while the support service can suggest modules for the better diagnostics.

Reporting issues

1. Open the PhpStorm tracking system at https://youtrack.jetbrains.com/ .
2. Click Create issue.
3. On the page that opens, choose PhpStorm from the Project drop-down list.
4. Describe your problem and provide a brief summary of it in the Description and Summary fields respectively.
5. If necessary, attach a screenshot that illustrates your problem.
6. Click Create issue when ready.

Sharing feedback

To share your feedback, do one of the following:

- Choose Help | Submit Feedback, which redirects you to the online feedback form.
  This form enables you to create a PhpStorm-specific YouTrack issue.

  The landing page depends on the PhpStorm version:

  - For release versions, the menu item opens the http://www.jetbrains.com/feedback/feedback.jsp page.
  - For EAP builds , the menu item opens the http://confluence.jetbrains.com/display/PhpStorm/PhpStorm+Early+Access+Program page.

Seeking assistance

To find assistance, do one of the following:

- Apply to the JetBrains Support
- Write to the support service. Use the following address:
  - intellij-support@jetbrains.com
  - phpstorm-support@jetbrains.com

  If necessary, attach the source code and the PhpStorm log .

  If necessary, attach the summary of PhpStorm settings related to your problem:

  1. On the main menu, choose Help | Settings Summary. The Settings Summary dialog box opens.
  2. From the Issue Type drop-down list, choose the subsystem in which you encountered a problem, the available options are:
     - Web Debug
     - CLI Debug
     - Deployment to view the settings that affect interaction with remote hosts.
     - PHPUnit to view the settings that affect unit testing.

     Depending on your choice, PhpStorm displays all the settings that are related to or may affect the activities in the chosen area.
  3. To save the summary to clipboard, click Copy.

- Ask the PhpStorm community .

Keymap Reference

PhpStorm provides the default platform-specific keymap reference in `pdf` format.

To view the keymap reference, choose Help | Keymap Reference on the main menu.

Sending Feedback

Your feedback, including error reports, improvement suggestions, new feature requests and any other things you might have to say to JetBrains team, is welcome at the addresses listed below.

- JetBrains support
- Community support