# XI – Provajderi sadržaja

## S A D R Ž A J

# 11.1-Uvod u Android deljenje podataka

- Sve Android aplikacije rade unutar svog vlastitog okruženja – ***sandbox***
- Time je onemogućeno da jedna aplikacija pristupa podacima koji pripadaju nekoj drugoj aplikaciji
- Da bi se to omogućilo potrebno je da se deklarišu potrebna ovlašćenja
- Android OS predlaže korišćenje **provajdera sadržaja** za deljenje podataka između različitih Android paketa.
- Za **provajder sadržaja** se može reći da predstavlja određeno skladište podataka kome paketi pristupaju primenom odgovarajućeg interfejsa.
- U većini slučajeva provajder sadržaja se ponaša slično bazi podataka.
- Moguće je postavljati upite, menjati, dodavati ili uklanjati podatke, itd.
- Međutim, za razliku od baze podataka, provajder sadržaja može da koristi različite načine skladištenja podataka.
- Podaci su smešteni u bazi podataka, datoteku ili dostupni preko mreže.
- Android poseduje poseban interfejs **ContentProvider** koji služi kao most između dve ili više aplikacija
- On jasno **razdvaja** sloj podataka od aplikacije i omogućava da aplikacije mogu međusobno da **dele i razmenjuju podatke**.

# 11.1-Uvod u Android deljenje podataka

- Za upotrebu **ContentProvider** potrebno je dati odgovarajuća ovlašćenja u datototeci **AndroidManifest.xml**
- Android ima nekoliko veoma korisnih ugrađenih provajdera sadržaja:
  1. ***Browser*** – podaci kao što su posećene Web stranice, istorija pregleda stranica ili traženja na Web-u, itd;
  2. ***CallLog*** – prikazivanje i ažuriranje istorije telefonskih poziva;
  3. ***Contacts*** – čitanje, menjanje i unošenje ličnih kontakta;
  4. ***MediaStore*** – čuva multimedijalne datoteke (audio,video, slike);
  5. ***LiveFolders***–specijalan folder kome pristupamo sa **ContentProvider**
  6. ***Settings*** – čuva podešavanja uređaja i preferencije korisnika.
  7. ***SearchRecentSuggestions*** - podešava se iz **ContentProvider**-a da čuva pojmove koje je korisnik nedavno pretraživao
  8. ***SearchRecentContract*** - preko objekta tipa **ContentProvider** povezuje podatke sa nizom podataka iz naloga
  9. ***UserDictinary*** – sadrži reči koje daje korisnik a upotrebljavaju ih metode za predviđanje teksta prilikom njegovog unosa
- Pored ugrađenih, Android podržava rad sa provajderima sadržaja koje su kreirali programeri prilikom razvoja Android aplikacija.

# 11.1-Uvod u Android deljenje podataka

- Da bi mogli da radimo sa **provajderom sadržaja** u Android OS, aplikaciji je potreban objekat tipa **ContentResolver**
- Zahvaljujući ovom objektu aplikacija može učitavati, umetati, brisati i ažurirati podatke u specificiranom izvoru sadržaja

```
// Pravljenje instance ContentProvider
ContentResolver crInstance = getContentResolver();
// Učitavanje podataka/kontakta iz provajdera sadržaja
crInstance.query(People.CONTENT_URI, null, null, null, null);
ContentValues new_Values= new ContentValues();
// Ubacivanje novih podataka/kontakta
crInstance.insert(People.CONTENT_URI, new_Values);
// Brisanje svih podataka/kontakta
crInstance.delete(People_URI, null, null);
ContentValues update_Values= new ContentValues();
// Ažuriranje vrednosti u provajderu sadržaja
crInstance.update(People_URI, update_Value, null,null);
```

# 11.1-Uvod u Android deljenje podataka

- U Android OS, upit nad provajderom sadržaja koristi formu **URI** (***Uniform Resource Identifier***) identifikatora sa opcionim specifikatorom koji se odnosi na konkretnu vrstu.
- Opšti oblik upita nad provajderom sadržaja izgleda ovako:

***<standardni_prefiks>://<vlasnik>/<putanja_podataka>/<id>***

- Upit je izgrađen iz sledećih komponenata:
  - ***standardni_prefiks*** - za provajdere sadržaja je uvek ***content://***.
  - ***vlasnik*** – predstavlja naziv provajdera sadržaja.
  - ***putanja_podataka*** – specificira vrstu traženih podataka. Na primer, ukoliko su u aplikaciji neophodni kontakti iz ***Contacts*** provajdera sadržaja, putanja može da bude označena kao ***people***, a **URI** identifikator da glasi: ***content://contacts/people***.
  - ***id*** – specificira zahtevani zapis.

**Primer:** *Ako se zahteva peti kontakt u* ***Contacts*** *provajderu sadržaja,* **URI** *identifikator može da ima sledeći oblik*:

***content://contacts/people/5***.

# 11.1-Uvod u Android deljenje podataka

- Na sledećoj slici predstavljeni su neki od često korišćeni stringova upita nad provajderima sadržaja.

| String upita | Opis |
|---|---|
| content://*media/internal/images* | Vraća listu svih slika iz interne memorije |
| content://*media/external/images* | Vraća listu svih slika iz eksterne memorije |
| content://*call_log/calls* | Vraća listu svih poziva koje je registrovao Call Log |
| content://*browser/bookmarks* | Vraća listu zapamćenih stranica web čitačem |

# 11.2 – Primer

- Za razumevanje koncepta provajdera sadržaja biće kreiran projekat pod nazivom ***Provider***, a njegova **main.xml** datoteka ima sledeći kod:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<ListView
    android:id="@+id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:stackFromBottom="false"
    android:transcriptMode="normal" />

<TextView
    android:id="@+id/contactName"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/contactID"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

# 11.2 - Primer

- **AndroidManifest.xml** datoteka sa ugrađenom dozvolom pristupa odgovarajućem sadržaju, data je sledećim kodom:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Provider"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.READ_CONTACTS"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".ProviderActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# 11.2 – Primer

- JAVA klasom aktivnosti projekta obavljaju se sve aktivnosti vezane za pristup i manipulaciju sadržajem koji obezbeđuje određeni provajder
- **URI** objekat, koji je odgovoran za izvršavanje stringa upita, ugrađen je u klasu aktivnosti aplikacije.
- Sledećim kodom je predstavljena klasa aktivnosti projekta ***Provider***.

```
package net.learn2develop.Provider;
import android.app.ListActivity;
public class ProviderActivity extends ListActivity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = Uri.parse("content://contacts/people");

        Cursor c;
        if (android.os.Build.VERSION.SDK_INT <11) {
            //---pre verzije Honeycomb---
            c = managedQuery(allContacts, null, null, null, null);
            //---Dozvoljava da aktivnost upravlja kursorom
            startManagingCursor(c);

        } else {
            //---Honeycomb i novije verzije---
            CursorLoader cursorLoader = new CursorLoader(
            this,allContacts,null,null,null,null);
            c = cursorLoader.loadInBackground();
        }
```

```
import android.app.ListActivity;
import android.content.CursorLoader;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.CursorAdapter;
import android.widget.SimpleCursorAdapter;
import android.util.Log;
```

```
        String[] columns = new String[] {
            ContactsContract.Contacts.DISPLAY_NAME,
            ContactsContract.Contacts._ID};

        int[] views = new int[] {R.id.contactName, R.id.contactID};

        SimpleCursorAdapter adapter;

        if (android.os.Build.VERSION.SDK_INT <11) {
            //---pre Honeycomb---
            adapter = new SimpleCursorAdapter(
                    this, R.layout.main, c, columns, views);
        } else {
            //---Honeycomb i novije verzije---
            adapter = new SimpleCursorAdapter(
                    this, R.layout.main, c, columns, views,
                    CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
        }
        this.setListAdapter(adapter);}

}
```

# 11.2 – Primer

- Postojanje podataka o kontaktima, u mobilnom telefonu ili emulatoru, uslov je da bi ti podaci mogli da budu prikazani.
- Iz tog razloga, neophodno je, ukoliko je lista kontakata prazna, dodati nekoliko kontakata u provajder sadržaja ***Contacts***.
- Aplikacija učitava sve kontakte iz provajdera sadržaja ***Contacts***, a zatim ih prikazuje primenom ***ListView*** pogleda.
- Za pristup provajderu ***Contacts***, kreiran je **URI** upit:

  **Uri allContacts = Uri.*parse("content://contacts/people");***
- Dalje se proverava verzija Androida, na kome se aplikacija izvršava.
- Ako je OS stariji od verzije Honeycomb (***Android API < 11***) moguće je koristiti ***menageQuery()*** metodu za manipulisanje kursorom koji rukuje svim događajima koji se odnose na pauziranje i restartovanje aplikacija, dok novije verzije koriste ***CursorLoader*** klasu:

```
CursorLoader cursorLoader = new CursorLoader(
this,allContacts,null,null,null,null);
c = cursorLoader.loadInBackground();
```

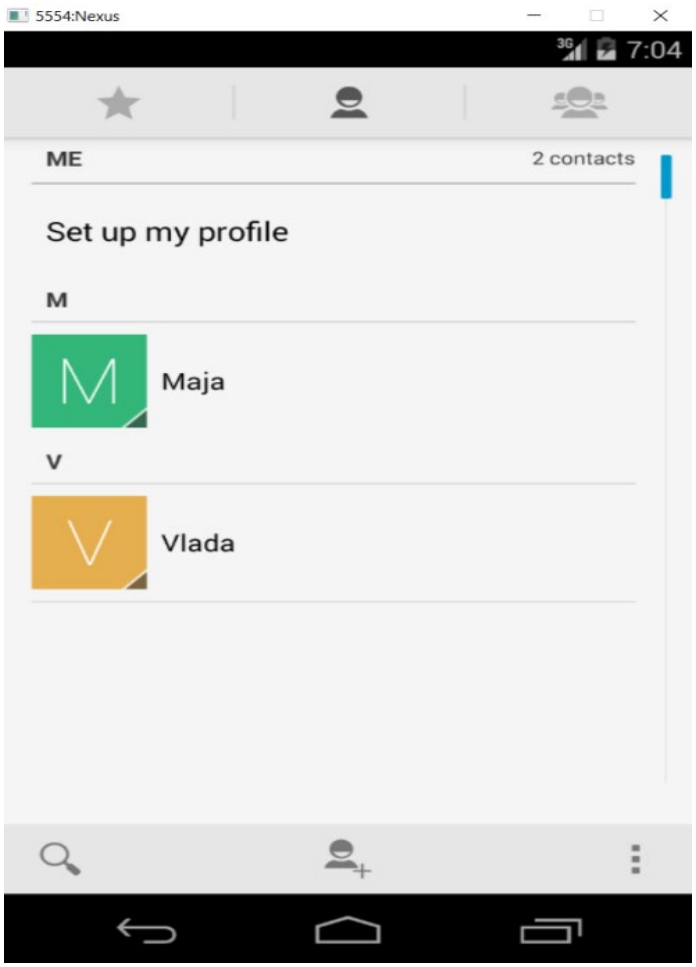
# 11.2 – Primer

- Ova klasa izvršava upit uz korišćenje kursora u pozadinskoj niti i na taj način ne blokira korisnički interfejs aplikacije.
- Objekat klase ***SimpleCursorAdapter*** povezuje **TextView** (ili **ImageView**) poglede definisane u **main.xml** datoteci.
- Prikazani kod predstavlja i prevaziđeni stari konstruktor ovog objekat i novi koji se koristi u novijim verzijama Androida (API nivo 11 i veći)
- Novi konstruktor koristi ***Fleg*** za registrovanje adaptera i na taj način dobija informacije o promenama na strani provajdera sadržaja.
- Takođe, aplikacija zahteva ***READ_CONTACTS*** privilegiju, u **AndroidManifest.xml** datoteci, da bi mogla da pristupi sadržaju provajdera ***Contacts***.

```
//---Honeycomb i novije verzije---
adapter = new SimpleCursorAdapter(
        this, R.layout.main, c, columns, views,
        CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
```

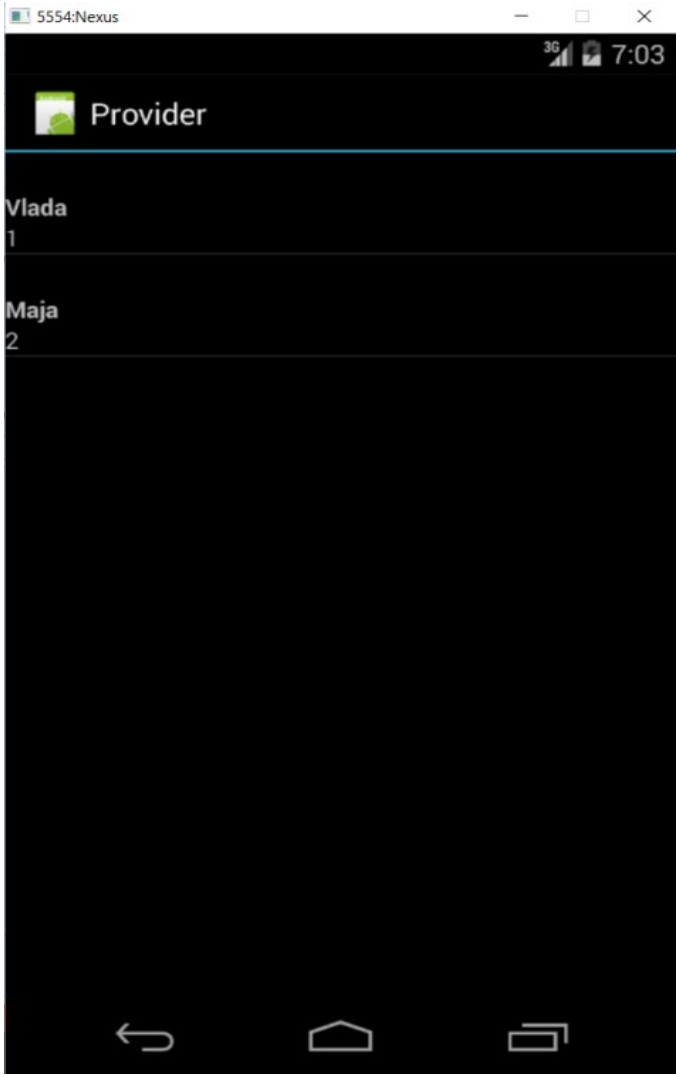
# 11.2 – Primer

- Klikom na F11, aplikacija se prevodi i pokreće emulatorom.
- Ukoliko je lista kontakta prazna, neophodno je uneti nekoliko kontakta primenom aplikacije ***Contacts***



# 11.2 – Primer

- Pokretanjem aplikacije emulatorom, ili Android telefonom, prikazuje se lista svih kontakta iz provajdera ***Contacts***



# 11.3 - Primena ugrađenih konstanti

- Pored URI identifikatora upita, moguće je koristiti i listu ugrađenih konstanti stringa upita za specificiranje **URI** identifikatora, za različite tipove podataka, u Android aplikacijama.
- Na primer, sledeće dve naredbe su ekvivalentne:

  **Uri allContacts = Uri.*parse("content://contacts/people");***

  **Uri allContacts = ContactsContract.Contacts.*CONTENT_URI*;**
- Slede primeri najčešće korišćenih konstanti za obraćanje provajderima sadržaja:
  - ***Browser.BOOKMARKS_URI;***
  - ***Browser.SEARCHES_URI;***
  - ***CallLOg.CONTENT_URI;***
  - ***MediaStore.Images.Media.INTERNAL_CONTENT_URI;***
  - ***MediaStore.Images.Media.EXTERNAL_CONTENT_URI;***
  - ***Settings_CONTENT_URI.***
- Za očitavanje prvog kontakta, identifikacioni broj se specificira na sledeći način:

  **Uri allContacts = Uri.*parse("content://contacts/people/1");***

# 11.3 - Primena ugrađenih konstanti

- Kao alternativu, moguće je koristiti predefinisanu konstantu sa metodom ***withAppendedID()*** klase ***ContentUris***:

**Uri allContacts=ContentUris.withAppendedID(ContactsContract.Contacts.*CONTENT_URI*,1)**

- Podatke je moguće,umesto **ListView** pogledom,prikazivati i kursorom:

```
package net.learn2develop.Provider;
import android.app.ListActivity;                    //ovde je uključeno i android.util.log
public class ProviderActivity extends ListActivity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = ContactsContract.Contacts.CONTENT_URI;

        Cursor c;
       *****
           }
           this.setListAdapter(adapter);
           PrintContacts(c);

    }

private void PrintContacts(Cursor c)
{
    if (c.moveToFirst()) {
        do{
        String contactID = c.getString(c.getColumnIndex(
                ContactsContract.Contacts._ID));
        String contactDisplayName =
                c.getString(c.getColumnIndex(
                        ContactsContract.Contacts.DISPLAY_NAME));
            Log.v("Content Providers", contactID + ", " +
                    contactDisplayName);
        } while (c.moveToNext());
    }
}

}
```

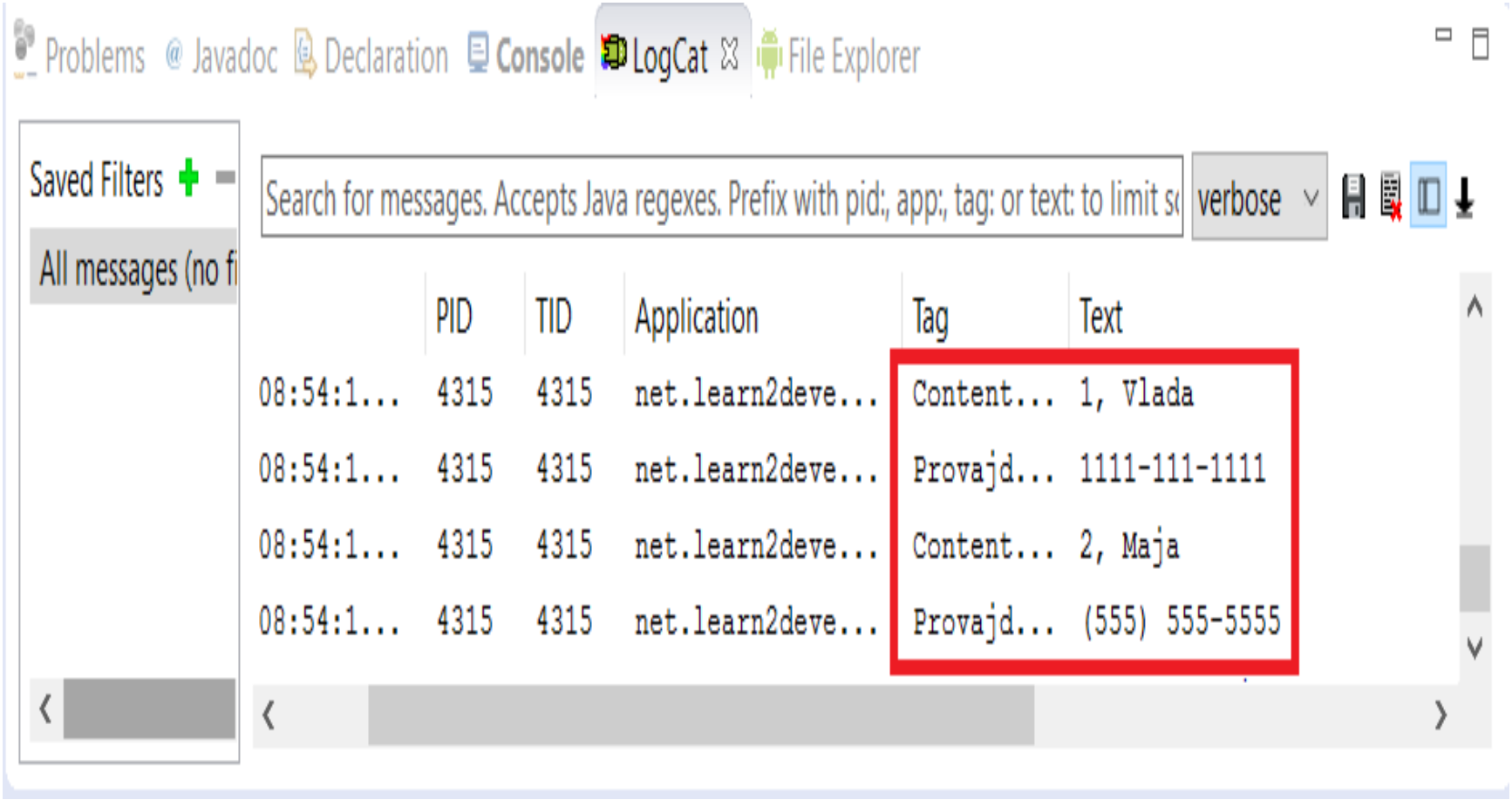
# 11.3 - Primena ugrađenih konstanti

- Prethodnim primerom, preuzete su informacije koje se odnose na identifikacioni broj i naziv svakog kontakta iz aplikacije ***Contacts***.
- Ukoliko se želi preuzimanje još neke informacije, telefonskog broja, neophodno je još jednom izvršiti upit nad provajderom sadržaja

```
private void PrintContacts(Cursor c){
    if (c.moveToFirst()) {
        do{
        String contactID = c.getString(c.getColumnIndex(
                ContactsContract.Contacts._ID));
        String contactDisplayName =
                c.getString(c.getColumnIndex(
                        ContactsContract.Contacts.DISPLAY_NAME));
            Log.v("Content Providers", contactID + ", " +
                contactDisplayName);
            //učitavanje broja telefona
            int hasPhone=
                    c.getInt(c.getColumnIndex(
                            ContactsContract.Contacts.HAS_PHONE_NUMBER));
            if (hasPhone==1){
                Cursor phoneCursor =
                getContentResolver().query(
                ContactsContract.CommonDataKinds.Phone.CONTENT_URI,null,
                ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " +
                contactID, null, null);
                while (phoneCursor.moveToNext()){
                    Log.v("Provajderi sadržaja ", phoneCursor.getString(
                            phoneCursor.getColumnIndex(
                            ContactsContract.CommonDataKinds.Phone.NUMBER)));
                }
                phoneCursor.close();
            }
        } while (c.moveToNext());
    }
}
```

# 11.3 - Primena ugrađenih konstanti

- Prethodni kod, sadržan u proširenoj metodi ***PrintContacts()***, prvo proverava da li kontakt sadrži telefonski broj primenom polja ***ContactsContract.Contacts.HAS_PHONE_NUMBER.***
- Ukoliko kontakt sadrži bar jedan telefonski broj, upit nad provajderom sadržaja ***Contacts*** biće ponovo izvršen i preuzeće se brojevi telefona



# 11.3 - Primena ugrađenih konstanti

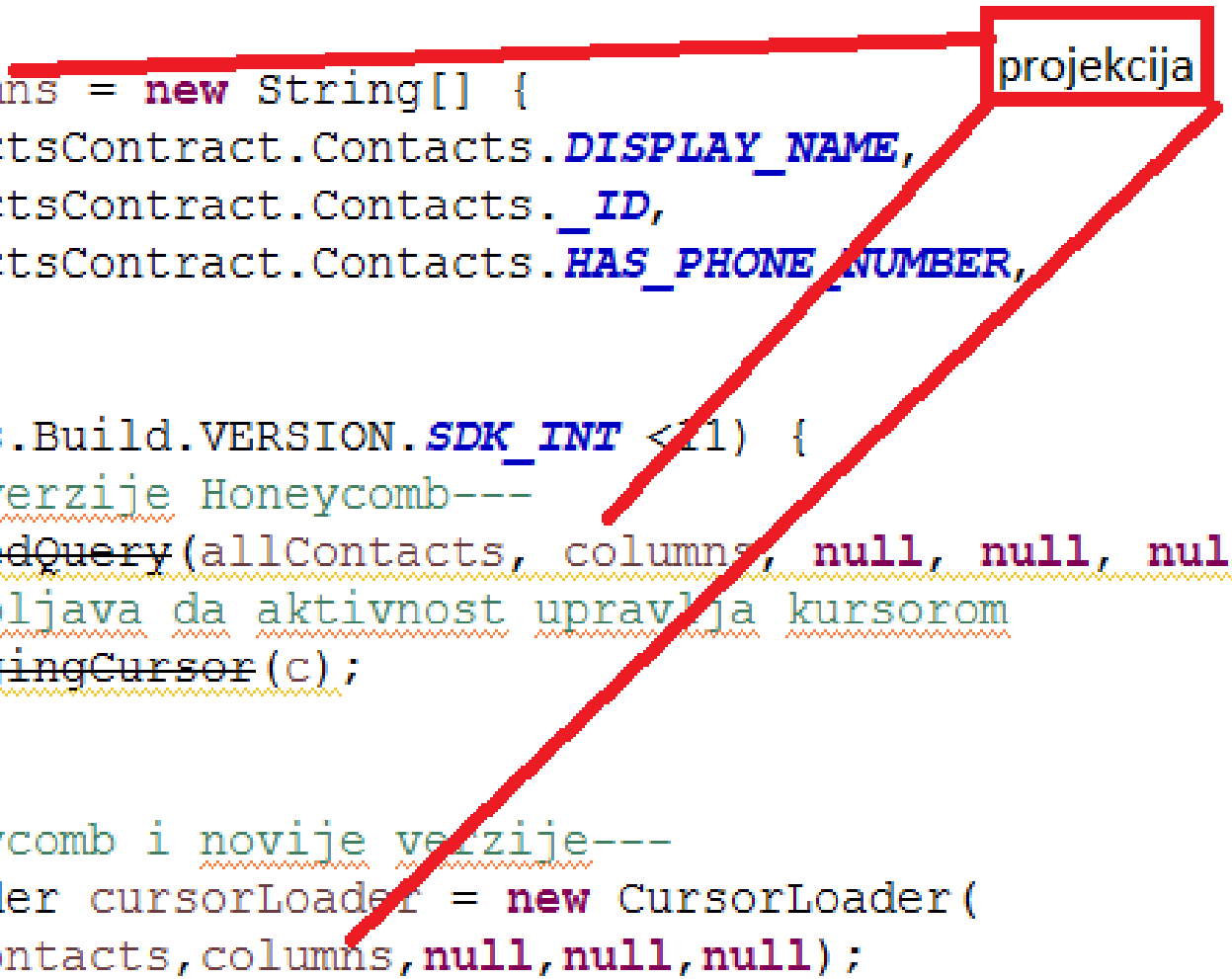
- U oba načina, primena metode ***manageQuery()*** je prevaziđena i koristi se aktuelna klasa ***CursorLoader***, koja koristi parametre kojim se određuje koliko kolona se vraća prilikom izvršavanja upita.
- Ovaj parametar se naziva **projekcija**.
- U prikazanom primeru, njegova vrednost iznosi ***null*** :

```
Cursor c;
if (android.os.Build.VERSION.SDK_INT <11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, null, null, null, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);

} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
    this,allContacts,null,null,null,null);
    c = cursorLoader.loadInBackground();
}
```

# 11.3 - Primena ugrađenih konstanti

- Ovakva manipulacija podacima, omogućava da se tačno specificira broj kolona koje se vraćaju kada kreiramo polja sa nazivima kolona:

```
Cursor c;
String[] columns = new String[] {
        ContactsContract.Contacts.DISPLAY_NAME,
        ContactsContract.Contacts._ID,
        ContactsContract.Contacts.HAS_PHONE_NUMBER,

        };
if (android.os.Build.VERSION.SDK_INT <11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns, null, null, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);

} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
    this,allContacts,columns,null,null,null);
    c = cursorLoader.loadInBackground();
}
```



# 11.4 - Filtriranje podataka

- Filtriranje je omogućeno kroz izvršavanje SQL klauzule ***WHERE***, a to je određeno trećim i četvrtim parametrom prevaziđene stare metode ***manageQuery*** i četvrtim i petim parametrom aktuelnog pristupa koji podrazumeva korišćenje klase ***CursorLoader***.

**Primer**: *sledeći kod čita samo one kontakte koji počinu slovom **v**.*

```
Cursor c;
String[] columns = new String[] {
        ContactsContract.Contacts.DISPLAY_NAME,
        ContactsContract.Contacts._ID,
        ContactsContract.Contacts.HAS_PHONE_NUMBER,

        };
if (android.os.Build.VERSION.SDK_INT <11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns,
            ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
            new String[] {"V%"}, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);

} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
    this,allContacts,columns,ContactsContract.Contacts.DISPLAY_NAME +
    " LIKE ?",
    new String[] {"V%"},null);
    c = cursorLoader.loadInBackground();
}
```

# 11.4 - Filtriranje podataka

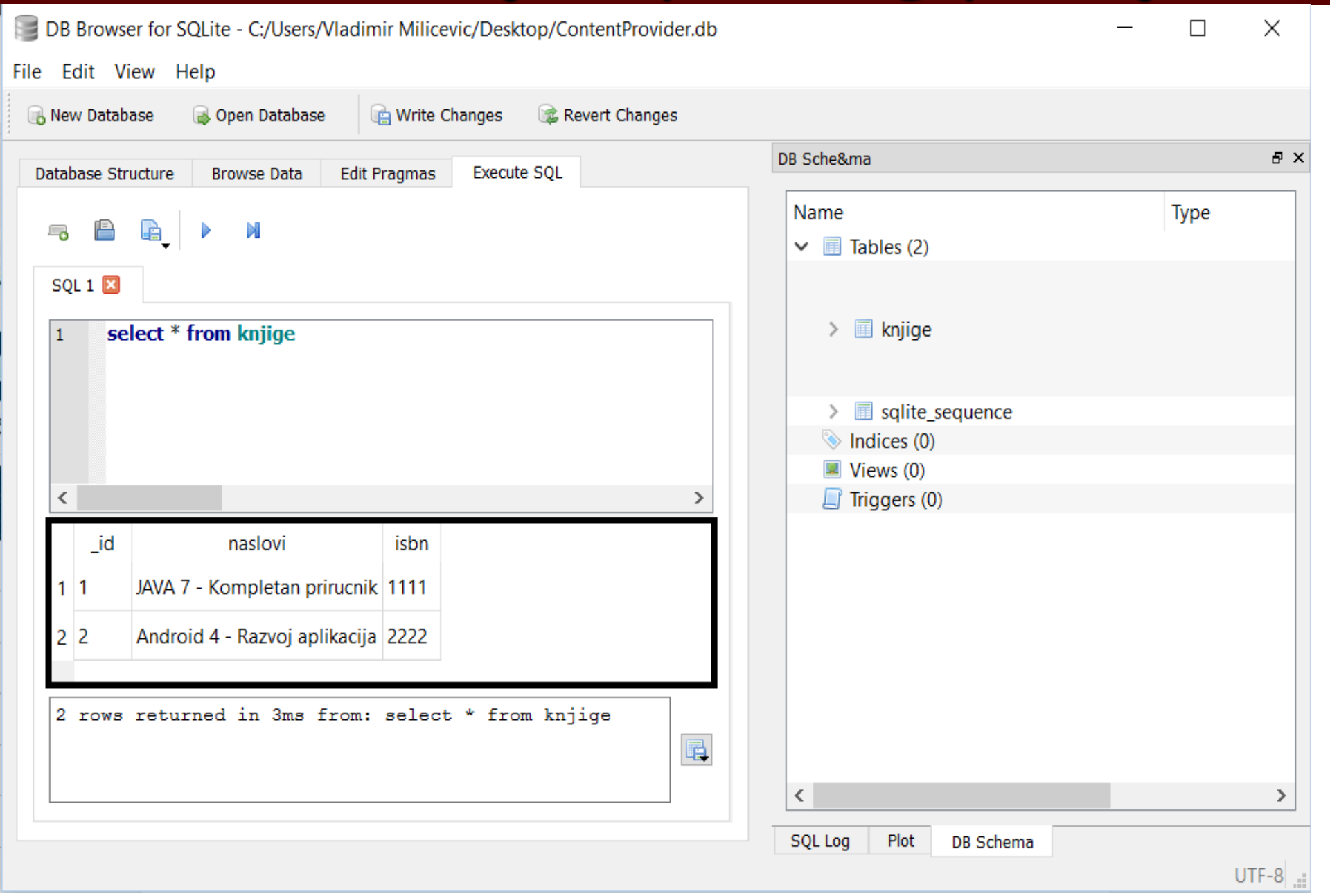
- Poslednji parametar, kod oba pristupa, omogućava specificiranje **SQL** klauzule ***ORDER BY*** kojom se realizuje sortiranje rezultata i izvršavanja upita koje je prikazano kodom na donjoj slici:

```
Cursor c;
String[] columns = new String[] {
        ContactsContract.Contacts.DISPLAY_NAME,
        ContactsContract.Contacts._ID,
        ContactsContract.Contacts.HAS_PHONE_NUMBER,

        };
if (android.os.Build.VERSION.SDK_INT <11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns,
            ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
            new String[] {"V%"},
            ContactsContract.Contacts.DISPLAY_NAME + " ASC");
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);

} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
    this,allContacts,columns,ContactsContract.Contacts.DISPLAY_NAME +
    " LIKE ?",
    new String[] {"V%"},
    ContactsContract.Contacts.DISPLAY_NAME + " ASC");
    c = cursorLoader.loadInBackground();
}
```

# 11.5 – Kreiranje sopstvenog provajdera

- Kreiranje vlastitog provajdera sadržaja je u osnovi veoma jednostavno.
- Neophodno je implementirati novu klasu, koja nasleđuje iz apstraktne klase ***ContentProvider*** i zatim definisati različite metode te klase:
  - ***query()*** - Omogućava spoljašnjim aplikacijama da učitavaju sadržaj;
  - ***insert()***-Omogućava spoljašnjim aplikacijama da ubacuju novi sadržaj;
  - ***update()*** - Omogućava spoljašnjim aplikacijama da ažuriraju sadržaj;
  - ***delete()*** - Omogućava spoljašnjim aplikacijama da brišu sadržaj;
  - ***getType()*** - Omogućava spoljašnjim aplikacijama da učitavaju sve podržane URI strukture;
  - ***onCreate()*** - Pravi instancu na bazu podataka iz koje se učitavaju podaci(sadržaj);
- Kao primer, biće kreiran novi provajder sadržaja koji skladišti knjige u tabeli baze podataka.
- Tabela će sadržati samo tri polja i to: ***_id, naslov*** i ***isbn.***
- Sledećom slikom prikazana je tabela baze podataka u kojoj će novo kreirani **provajder sadržaja** skladištiti knjige.

# 11.5 - Kreiranje sopstvenog provajdera



# 11.5 - Kreiranje sopstvenog provajdera

- U prvom koraku biće kreirana klasa provajdera sadržaja sa pomoćnom klasom ***DataBaseHelper*** i mehanizmima za kreiranje baze podataka:

```
package net.learn2develop.ContentProviders;
import android.content.ContentProvider;*
public class BooksProvider extends ContentProvider{
    static final String PROVIDER_NAME =
        "net.learn2develop.provider.Books";

    static final Uri CONTENT_URI =
        Uri.parse("content://"+ PROVIDER_NAME + "/books");

    static final String _ID = "_id";
    static final String TITLE = "naslovi";
    static final String ISBN = "isbn";

    static final int BOOKS = 1;
    static final int BOOK_ID = 2;

    private static final UriMatcher uriMatcher;
    static{
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
        uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
    }
```

```
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;
```

```
    //---kreiranje baze podataka---
    SQLiteDatabase booksDB;
    static final String DATABASE_NAME = "Books";
    static final String DATABASE_TABLE = "knjige";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE =
        "create table " + DATABASE_TABLE +
        " (_id integer primary key autoincrement, "
        + "naslovi text not null, isbn text not null);";
    private static class DatabaseHelper extends SQLiteOpenHelper{
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db){
            db.execSQL(DATABASE_CREATE);
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion,
                int newVersion) {
            Log.w("Baza podataka provajera sadržaja",
                    "Ažuriranje verzije sa " +
                            oldVersion + " na verziju " + newVersion +
                    ", stari podaci biće uništeni");
            db.execSQL("DROP TABLE IF EXISTS titles");
            onCreate(db);
        }
    }
}
```

# 11.5 - Kreiranje sopstvenog provajdera

- Pored podrške za kreiranje baze podataka, u klasu provajdera sadržaja je neophodno uneti metode koje obezbeđuju izvođenje operacija nad bazom podataka: ***getType()***, ***onCreate()*** i ***query()***.

```
@Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)){
        //---get all books---
        case BOOKS:
            return "vnd.android.cursor.dir/vnd.learn2develop.books ";

        //---get a particular book---
        case BOOK_ID:
            return "vnd.android.cursor.item/vnd.learn2develop.books ";

        default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
    }
@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    booksDB = dbHelper.getWritableDatabase();
    return (booksDB == null)? false:true;
}
```

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
    sqlBuilder.setTables(DATABASE_TABLE);

    if (uriMatcher.match(uri) == BOOK_ID)
        //---ako se pruzima konkretna knjiga---
        sqlBuilder.appendWhere(
                _ID + " = " + uri.getPathSegments().get(1));

    if (sortOrder==null || sortOrder=="")
        sortOrder = TITLE;

    Cursor c = sqlBuilder.query(
        booksDB,
        projection,
        selection,
        selectionArgs,
        null,
        null,
        sortOrder);
    //---registrovanje praćenja promena URI identifikatora---
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}
```

# 11.5 - Kreiranje sopstvenog provajdera

- Neophodno je dodati i metode ***insert()***, ***delete()*** i ***update()***.

```
@Override
    public int delete(Uri arg0, String arg1, String[] arg2) {
        int count=0;
        switch (uriMatcher.match(arg0)){
        case BOOKS:
            count = booksDB.delete(
                    DATABASE_TABLE,
                    arg1,
                    arg2);
            break;
        case BOOK_ID:
            String id = arg0.getPathSegments().get(1);
            count = booksDB.delete(
                    DATABASE_TABLE,
                    _ID + " = " + id +
                    (!TextUtils.isEmpty(arg1) ? " AND (" +
                            arg1 + ')' : ""),
                            arg2);
            break;
        default: throw new IllegalArgumentException("Unknown URI " + arg0);
        }
        getContext().getContentResolver().notifyChange(arg0, null);
        return count;
    }
```

```
@Override
    public Uri insert(Uri uri, ContentValues values) {
        //---dodaje novu knjigu---
        long rowID = booksDB.insert(DATABASE_TABLE,
                "",values);
        //---ako je dodavanje uspešno---
        if (rowID>0){
            Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
            getContext().getContentResolver().notifyChange(_uri, null);
            return _uri;
        }
        throw new SQLException("Dodavanje u vrstu nije uspelo " + uri);
    }
```

```
@Override
    public int update(Uri uri, ContentValues values, String selection,
            String[] selectionArgs) {
        int count = 0;
        switch (uriMatcher.match(uri)){
        case BOOKS:
            count = booksDB.update(
                    DATABASE_TABLE,
                    values,
                    selection,
                    selectionArgs);
            break;
        case BOOK_ID:
            count = booksDB.update(
                    DATABASE_TABLE,
                    values,
                    _ID + " = " + uri.getPathSegments().get(1) +
                    (!TextUtils.isEmpty(selection) ? " AND (" +
                            selection + ')' : ""),
                            selectionArgs);
            break;
        default: throw new IllegalArgumentException("Nepoznat URI " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }
```

# 11.5 - Kreiranje sopstvenog provajdera

- Da bi rad sa provajderom sadržaja bio omogućen, pored kreiranja klase provajdera sadržaja i dodavanja metoda, neophodno je napraviti i izvesne modifikacije u datoteci ***AndroidManifest.xml***.
- Koristeći XML tag ***<provider>*** ... ***</provider>***, ovom XML datotekom se uključuje novi provajder sadržaja.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.ContentProviders"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".ContentProvidersActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider android:name="BooksProvider"
            android:authorities="net.learn2develop.provider.Books">
        </provider>
    </application>
</manifest>
```

# 11.5 – Kreiranje sopstvenog provajdera

- Nakon prvog koraka, kreiranja klase naslednice osnovne klase ***ContentProvider***, predefinisano je nekoliko metoda bazne klase:
  - ***getType()*** – vraća MIME tip sa odgovarajućim URI;
  - ***onCreate()*** – izvršava se sa pokretanjem provajdera;
  - ***query()*** – učitava zahtev klijenta i vraća ***Cursor*** objekat;
  - ***insert()*** – unosi novi zapis u provajder sadržaja;
  - ***delete()*** – briše zapis pomoću provajdera sadržaja;
  - ***update()*** – koristi provajder sadržaja za ažuriranje zapisa.
- U inicijalnom kodu klase moguće je primetiti da je korišćen objekat ***UriMatcher*** za analiziranje sadržaja **URI identifikatora** koji je prosleđen provajderu objektom ***ContentResolver.***

**Primer**: *sledećim URI identifikatorima su zatraženi zahtevi za učitavanje svih knjiga i knjige čiji identifikator ima vrednost 1, respektivno.*

```
Uri.parse("content://"+ PROVIDER_NAME + "/books");
Uri.parse("content://"+ PROVIDER_NAME + "/books/1");
```

# 11.5 – Kreiranje sopstvenog provajdera

- Provajder koristi ***SQLite*** bazu podataka pa je iskorišćena ***SQLiteHelper*** pomoćna klasa za lakše upravljanje bazom podataka.
- Predefinisanjem metode ***getType()*** tako što mu se predaje ***URI*** objekat, dobijen je unikatan način za opisivanje tipa podataka za provajdera.
- Primenom ***UriMatcher*** objekta više knjiga se učitava pomoću: ***vnd.android.cursor.dir/vnd.learn2develop.books***, a pojedinačne knjige pomoću: ***vnd.android.cursor.item/vnd.learn2develop.books***
- U sledećem koraku predefinisana je ***onCreate()*** metoda sa ciljem omogućavanja konekcije sa bazom podataka nakon pokretanja provajdera sadržaja.
- Takođe, predefinisana je i ***query()*** metoda kojom je omogućeno klijentima da postavljaju upite za knjige.
- Metoda je podešena tako da se rezultat upita vraća kao tip ***Cursor***, sortiran po polju ***TITLE***.

# 11.5 – Kreiranje sopstvenog provajdera

- Da bi nov podatak bio unešen u bazu podataka, primenom provajdera sadržaja, neophodno je koristiti predefinisanu metodu ***insert()*** koja preuzima kao argumente dva objekta ***Uri*** i ***ContatntValues***.
- Kao rezultat, metoda vraća tip podataka ***Uri***.
- Nakon obavljenog dodavanja, novog zapisa u bazu, izvršava se metoda ***notifyChange()*** objekta klase ***ContentResolver***.
- Za uklanjanje zapisa, u konkretnom slučaju knjige, iz baze podataka, a primenom provajdera sadržaja, predefinisana je i upotrebljena metoda ***delete()***.
- Metoda ***delete()***, omogućava da se izvrši i metoda ***notifyChange()*** objekta klase ***ContentResolver*** nakon izvršenog uklanjanja podataka.
- Na ovaj način se obaveštavaju registrovani posmatrači da je obrisana odgovarajuća vrsta.
- U nastavku je predefinisana i iskorišćena metoda ***update()*** koja, slično kao i prethodne dve metode, izvršava metodu ***notifyChange()*** objekta klase ***ContentResolver***.

# 11.5 – Kreiranje sopstvenog provajdera

- Kroz ovu akciju obaveštavaju se registrovani posmatrači da je ažurirana odgovarajuća vrsta.
- Na samom kraju, da bi provajder sadržaja bio registrovan i angažovan u Android sistemu, modifikovana je datoteka **AndroidManifest.xml** dodavanjem **XML** elementa/taga ***<provider>*** .
- Za primenu kreiranog provajdera sadržaja, u Android aplikaciji, neophodno je kreirati klasu aktivnosti i korisnički interfejs aplikacije preko koga će korisnik i aplikacija komunicirati.
- Na sledećim slajdovima dat je kod odgovarajuće klase aktivnosti koja omogućava angažovanje našeg novo kreiranog provajdera sadržaja.
- Takođe, u folderu projekta, podfolder ***res/layout***, potrebno je izvršiti promene u datoteci **main.xml**
- Ovde treba dodati kod koji odgovara elementima korisničkog interfejsa i njihovom rasporedu na ekranu.
- Novom definicijom ove datoteke, kompletirana je aplikacija za demonstraciju primene vlastitog provajdera sadržaja.

# 11.5 - Kreiranje sopstvenog provajdera

```
package net.learn2develop.ContentProviders;
import android.app.Activity;
public class ContentProvidersActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClickAddTitle(View view) {
        /*
        //---add a book---
        ContentValues values = new ContentValues();
        values.put(BooksProvider.TITLE, ((EditText)
                findViewById(R.id.txtTitle)).getText().toString());
        values.put(BooksProvider.ISBN, ((EditText)
                findViewById(R.id.txtISBN)).getText().toString());
        Uri uri = getContentResolver().insert(
                BooksProvider.CONTENT_URI, values);
         */
        ContentValues values = new ContentValues();
        values.put("naslovi", ((EditText)
                findViewById(R.id.txtTitle)).getText().toString());
        values.put("isbn", ((EditText)
                findViewById(R.id.txtISBN)).getText().toString());
        Uri uri = getContentResolver().insert(
            Uri.parse("content://net.learn2develop.provider.Books/books"),
                    values);

        Toast.makeText(getBaseContext(),uri.toString(),
                Toast.LENGTH_LONG).show();
    }

    public void onClickRetrieveTitles(View view) {
        //---vraća naslov---
        Uri allTitles = Uri.parse("content://net.learn2develop.provider.Books/books");
        Cursor c;
        CursorLoader cursorLoader = new CursorLoader(this,allTitles, null, null,
                null,"naslovi desc");
        c = cursorLoader.loadInBackground();
        if (c.moveToFirst()) {
            do{
                Toast.makeText(this,
                        c.getString(c.getColumnIndex(
                                BooksProvider._ID)) + ", " +
                                c.getString(c.getColumnIndex(
                                        BooksProvider.TITLE)) + ", " +
                                        c.getString(c.getColumnIndex(
                                                BooksProvider.ISBN)),
                                                Toast.LENGTH_SHORT).show();
            } while (c.moveToNext());
        }
    }

    public void updateTitle() {
        ContentValues editedValues = new ContentValues();
        editedValues.put(BooksProvider.TITLE, "Android Tipovi i trikovi.");
        getContentResolver().update(
        Uri.parse("content://net.learn2develop.provider.Books/books/2"),
                        editedValues,null,null);
    }

    public void deleteTitle() {

        //---brisanje naslova---
        getContentResolver().delete(
                Uri.parse("content://net.learn2develop.provider.Books/books/2"),
                null, null);
        //---brisanje svih naslova---
        getContentResolver().delete(
                Uri.parse("content://net.learn2develop.provider.Books/books"),
                null, null);
    }

}
```

```
import android.app.Activity;
import android.content.ContentValues;
import android.content.CursorLoader;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
```

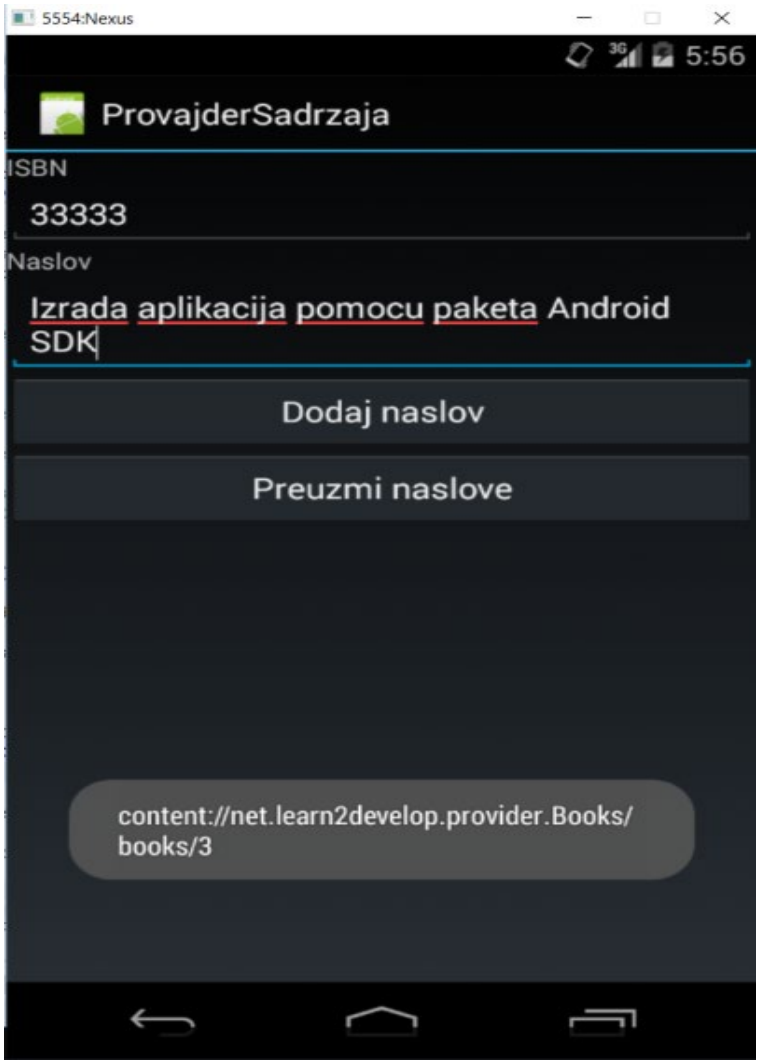
# 11.5 - Kreiranje sopstvenog provajdera

- Sledećim kodom data je main.xml datoteka:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="ISBN" />
<EditText
    android:id="@+id/txtISBN"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Naslov" />
<EditText
    android:id="@+id/txtTitle"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />
<Button
    android:text="Dodaj naslov"
    android:id="@+id/btnAdd"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickAddTitle" />
<Button
    android:text="Preuzmi naslove"
    android:id="@+id/btnRetrieve"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickRetrieveTitles"  />
</LinearLayout>
```

# 11.5 - Kreiranje sopstvenog provajdera

- Klikom na F11, program je preveden i pokrenut emulatorom.
- Pokreće se aktivnost koja je modifikovana tako da korisniku omogućava da unese ***ISBN*** broj i naslov knjige pomoću prethodno kreiranog provajdera sadržaja.



# 11.5 – Kreiranje sopstvenog provajdera

- Da bi provajder sadržaja dodao novu knjigu, neophodno je kreirati, u metodi ***onClickAddTitle()***, objekat tipa ***ContentValues*** u koji se pakuju informacije koje se odnose na konkretnu knjigu
- Kada se pogleda priloženi kod klase aktivnosti, za navedenu metodu moguće je uočiti da se jedan deo koda pojavljuje obeležen oznakom za komentare, a ispod njega je aktivan kod.
- Oba koda imaju zadatak da ukažu na polja ***ISBN*** i ***naslovi*** sa razlikom da prvi blok koda koristi konstane ***BooksProvider.ISBN*** i ***BooksProvider.TITLE***, respektivno, za pristupanje poljima, ukoliko je provajder sadržaja kreiran u istom paketu kao aplikacija
- Drugi blok koda omogućava pristupanje provajderu sadržaja, koji ne mora da se nalazi u paketu aplikacije, direktnim navođenjem naziva polja i kompletnog URI identifikatora sadržaja.

# 11.5 - Kreiranje sopstvenog provajdera

- Razlike su prikazane sledećom kodom:

```
/*  provajder je u istom paketu
//---add a book---
ContentValues values = new ContentValues();
values.put(BooksProvider.TITLE, ((EditText)
        findViewById(R.id.txtTitle)).getText().toString());
values.put(BooksProvider.ISBN, ((EditText)
        findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
        BooksProvider.CONTENT_URI, values);  */
```

```
//provajder ne mora da bude u istom paketu
ContentValues values = new ContentValues();
values.put("naslovi", ((EditText)
        findViewById(R.id.txtTitle)).getText().toString());
values.put("isbn", ((EditText)
        findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
    Uri.parse("content://net.learn2develop.provider.Books/books"),
                values);
```

# 11.5 - Kreiranje sopstvenog provajdera

- Angažovanjem metode ***onClickRetrieveTitles()***, omogućeno je učitavanje knjiga koje su prethodno definisane provajderom sadržaja.
- Takođe, u metodu je ugrađen upit koji kao rezultat vraća niz knjiga sortiran po polju ***title*** u opadajućem poretku (***naslovi desc***).

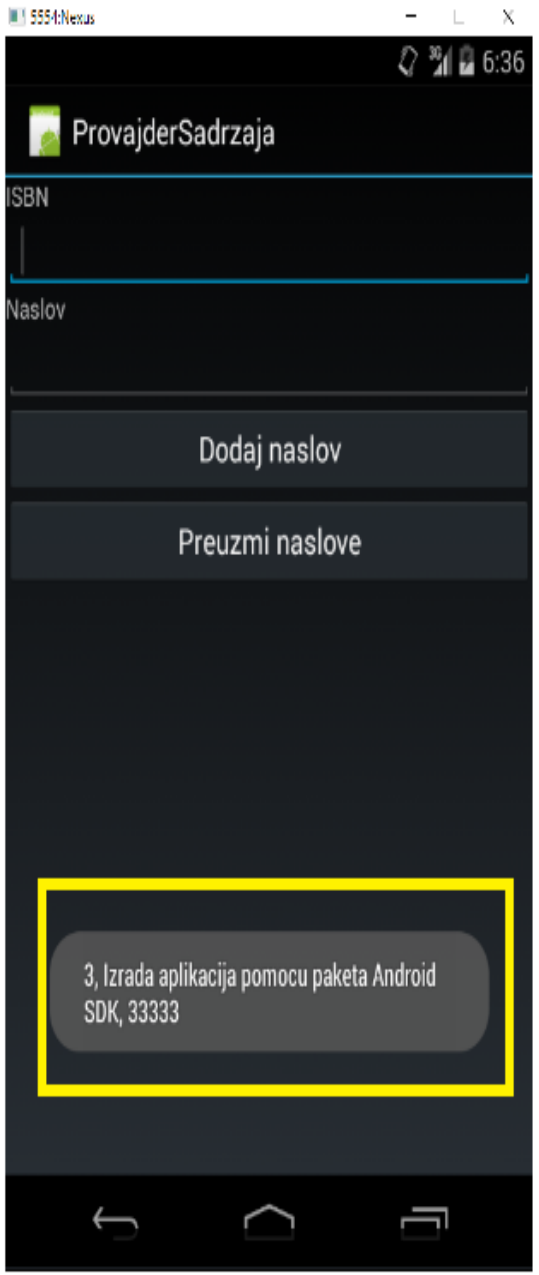
```
public void onClickRetrieveTitles(View view) {
    //---vraća naslov---
    Uri allTitles = Uri.parse("content://net.learn2develop.provider.Books/books");
    Cursor c;
    CursorLoader cursorLoader = new CursorLoader(this,allTitles, null, null,
            null,"naslovi desc" ;
        c = cursorLoader.loadInBackground();
    if (c.moveToFirst()) {
        do{
            Toast.makeText(this,
                    c.getString(c.getColumnIndex(
                            BooksProvider._ID)) + ", " +
                            c.getString(c.getColumnIndex(
                                    BooksProvider.TITLE)) + ", " +
                                    c.getString(c.getColumnIndex(
                                            BooksProvider.ISBN)),
                                            Toast.LENGTH_SHORT).show();
        } while (c.moveToNext());
    }
}
```

# 11.5 – Kreiranje sopstvenog provajdera

- Preuzete knjige prikazuju se na ekranu uređaja (na sledećem slajdu).
- Takođe, aplikacija omogućava da se ažururiraju detalji, koji se odnose na konkretnu knjigu, primenom metode ***update()*** klase aktivnosti aplikacije.
- Navođenjem **URI** identifikatora sadržaja ukazuje se na identifikator konkretne knjige (obeleženo crvenom bojom na sledećem slajdu).
- Takođe, primenom URI identifikatora sadržaja, metodom ***delete()***, omogućeno je brisanje pojedinačnih ili svih knjiga (obeleženo crnom bojom na sledećem slajdu).

# 11.5 - Kreiranje sopstvenog provajdera

```
public void updateTitle() {
    ContentValues editedValues = new ContentValues();
    editedValues.put(BooksProvider.TITLE, "Android Tipovi i trikovi.");
    getContentResolver().update(
    Uri.parse("content://net.learn2develop.provider.Books/books/2"),
                editedValues,null,null);
}

public void deleteTitle() {

    //---brisanje naslova---
    getContentResolver().delete(
            Uri.parse("content://net.learn2develop.provider.Books/books/2"),
            null, null);
    //---brisanje svih naslova---
    getContentResolver().delete(
            Uri.parse("content://net.learn2develop.provider.Books/books"),
            null, null);
}
```



# Hvala na pažnji !!!

Pitanja

? ? ?

# Lab4:Izrada sopstvenog izvora sadržaja

- Prikazaćemo kako možemo kreirati svoje izvore podataka koji će biti dostupni i drugim aplikacijama da ih koriste
- Kreiraćemo namenski izvor podataka tako što čemo proširiti Android klasu ContentProvider koja sadrži šest metoda koje možemo menjati:
  - ***query()-*** Omogućava spoljašnjim aplikacijama da učitavaju sadržaj;
  - ***insert()-***Omogućava spoljašnjim aplikacijama da ubacuju novi sadržaj;
  - ***update() -*** Omogućava spoljašnjim aplikacijama da ažuriraju sadržaj;
  - ***delete()*** – Omogućava spoljašnjim aplikacijama da brišu sadržaj;
  - ***getType()*** – Omogućava spoljašnjim aplikacijama da učitavaju sve podržane URI strukture;
  - ***onCreate()*** – Pravi instancu na bazu podataka iz koje se učitavaju podaci(sadržaj);
- U zavisnosti od aplikacije vrši se modifikacija potrebnih metoda
- U našem primeru izvršena je modifikacija metoda on Create() kako bi pristupili bazi MyDB kao I metode query() kako bi učitali te podatke
- Objektu **UriMatcher** dodajemo **URI** koji se formira od imena projekta (***com.cookbook.datastorage***) i imena tabele u bazi podataka **diaries**

# Lab4:Izrada sopstvenog izvora sadržaja

```
package com.cookbook.datastorage;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import com.cookbook.data.Constants;
import com.cookbook.data.MyDB;
public class DiaryContentProvider extends ContentProvider {
private MyDB dba;
private static final UriMatcher sUriMatcher;
//the code returned for URI match to components
private static final int DIARIES=1;
public static final String AUTHORITY = "com.cookbook.datastorage";
static {
   sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
   sUriMatcher.addURI(AUTHORITY, Constants.TABLE_NAME, DIARIES);
}
```

# Lab4:Izrada sopstvenog izvora sadržaja

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
return 0;
}
public String getType(Uri uri) {return null;}
public Uri insert(Uri uri, ContentValues values) {return null;}
public int update(Uri uri, ContentValues values, String selection,
                        String[] selectionArgs) {return 0;}
@Override
public boolean onCreate() {
        dba = new MyDB(this.getContext());
        dba.open();
        return false;
}
@Override
public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
  Cursor c=null;
  switch (sUriMatcher.match(uri)) {
    case DIARIES:
      c = dba.getdiaries();
      break;
    default:
      throw new IllegalArgumentException("Unknown URI " + uri);
  }
c.setNotificationUri(getContext().getContentResolver(), uri);
return c;
}
{
```

# Lab4: Fajl AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.cookbook.datastorage"
  android:versionCode="1"
  android:versionName="1.0">
<application android:icon="@drawable/icon"
  android:label="@string/app_name">
<activity android:name=".DataStorage"
  android:label="@string/app_name">
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".MyPreferences" />
<activity android:name=".Diary"/>
<activity android:name=".DisplayDiaries"/>
<provider android:name="DiaryContentProvider"
                    android:authorities="com.cookbook.datastorage" />
</application>
<uses-sdk android:minSdkVersion="7" />
</manifest>
```

# Lab4:Izrada sopstvenog izvora sadržaja

- Sada je provajder sadržaja spreman da ga koriste i druge aplikacije
- Da bi to testirali napravićemo novu Android aplikaciju DataStorage Tester koja će imati samo jednu aktivnost DataStorageTester
- Podatke čemo učitati iz provajdera sadržaja koristeći objekat tipa ContentResolver
- Nakon što formiramo objekat tipa Cursor, funkcija koja testira provajder sadržaja izvršiće izdvajanje svakog drugog polja iz svakog učitanog zapisa
- Pomoću objekta StringBuilder taj izdvojeni podatak dodaće na kraj novoformiranog znakovnog niza koji će prikazati na ekranu

```
package com.cookbook.datastorage_tester;
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;
public class DataStorageTester extends Activity {
```

# Lab4:Izrada sopstvenog izvora sadržaja

```
TextView tv;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
tv = (TextView) findViewByld(R.id.output);
String myUri = "content://com.cookbook.datastorage/diaries";
Uri CONTENT_URI = Uri.parse(myUri);
//get ContentResolver instance
ContentResolver crInstance = getContentResolver();
Cursor c = crInstance.query(CONTENT_URI, null, null, null, null);
startManagingCursor(c);
StringBuilder sb = new StringBuilder();
if(c.moveToFirst()){
do{
sb.append(c.getString(1)).append("\n");
}while(c.moveToNext());
}
tv.setText(sb.toString());
}
}
```

# Lab4:Sadržaj fajla res/layout/main.xml

- U datoteku sadržaja glavnog ekrana, **main.xml**, treba dodati ID natpisa koji će prikazivati rezultate na ekranu:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:id="@+id/output"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

# Lab4:Izrada sopstvenog izvora sadržaja