IX – Korisnički interfejs

S A D R Ž A J

9.1 – Pojam fragmenta

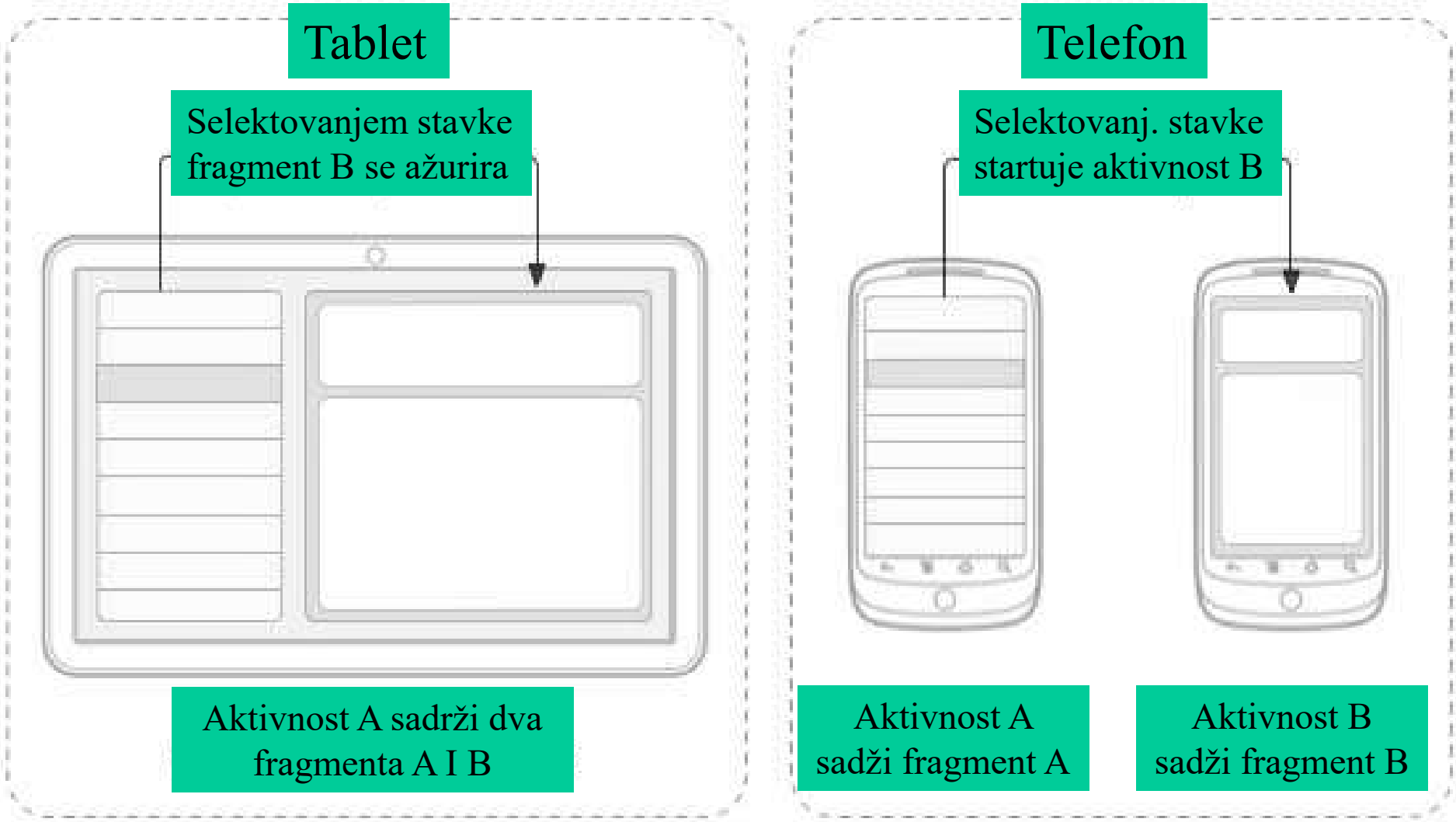
- Najveći problem kod aktivnosti je nemogućnost istovremenog prikaza više različitih UI događaja jer jedna aktivnost može prikazati samo jednu UI aktivnost na ekranu u jednom vremenskom trenutku.
- Javila se potreba za uvođenjem novog elementa koji će omogućiti veću fleksibilnost i ukloniti ograničenje prikaza samo jedne aktivnosti na ekranu istovremeno, odvojiti prikaz ekrana od kontrole različitih UI
- **Fragment** predstavlja deo korisničkog interfejsa Android aplikacije koji se može smestiti u aktivnost i time povećati njenu modularnost
- Sada imamo jednu aktivnost, ali svaka aktivnost može imati više fragmenata koji imaju svoj posebni tj. sopstveni životni ciklus.
- On predstavlja neku vrstu podaktivnosti u okviru glavne aktivnosti
- Fragment predstavlja način ponašanja dela UI interfejsa u aktivnosti.
- Fragmenti su gradivni elementi od kojih se sastoji jedna aktivnost.
- Najveća prednost fragmenata je što pojednostavljuje proces izrade prikaza za različite veličine ekrana, jer se svaki fragment prilagođava prostoru na ekranu uređaja
- Povećanje modularnosti, fleksibilnost i ponovne upotrebe delova koda.

9.1 – Karakteristike fragmenta

1. Svaki fragment ima svoje parametre: sopstveni raspored (*layout*), svoje događaje,ulaze i aktivnost koja je definisana svojim životnim ciklusom
2. Fragmenti su enkapsulirani, imaju svoju funkcionalnost, izgled i mogu reagovati na korisničke komande
3. Fragmenti se mogu posmatrati kao modularni delovi aktivnosti.
4. Fragment ne postoji bez aktivnosti već mora biti uključen u aktivnost
5. Fragmenti se mogu dodavati ili ukidati u jednoj aktivnosti dok je ta aktivnost aktivna tj. tek kada je ona pokrenuta
6. Možemo kombinovati više fragmenata u jednu aktivnost što omogućava aplikacije sa više različitih prikaza
7. Životni ciklus fragmenta je povezan sa matičnom aktivnosti što znači da kada je aktivnost zaustavljena zaustavljeni su i svi njeni fragmenti.
8. Međutim, ako pauziramo fragment, aktivnost nije pauzirana.
9. Svi fragmenti ne moraju da implementiraju tj. da imaju komponente korisničkog interfejsa
10. Za korištenje fragmenata je potrebno koristiti verziju Androida 3.0 (***Honeycomb***) ili veću (API verzija 11 ili više).

9.1 – Prednosti fragmenta

Primer: *kako možemo da prikažemo dve UI aktivnosti preko fragmenata koji mogu biti kombinovani u jednu aktivnost za tablet dizajn ili razdvojeni za dizajn kod telefona*



9.1 – Prednosti fragmenta

- **Fragmenti** imaju svoj životni ciklus ali su zavisni od životnog ciklusa Aktivnosti u kom se nalaze.
- Tako da ukoliko Aktivnost pređe u stanje pauze, i sami Fragmenti koji se nalaze u okviru te Aktivnosti prelaze u stanje pauze.
- Njima je moguće manipulisati, odnosno po potrebi ih postavljati i uklanjati sa Aktivnosti.
- Dodavanje ili uklanjanje Fragmenta predstavlja jednu transakciju.
- Ove transakcije je moguće pamtiti na steku o kome se brine Aktivnost, čime se pruža mogućnost uklanjanja poslednje dodatog Fragmenta pritiskom na dugme „nazad“, čime se simulira opcija skidanja poslednje dodatog elementa na stek.
- Fragmenti se kreiraju proširenjem **Fragment** klase
- Oni se ubacuju u aktivnost deklarisanjem u **layout** fajlu aktivnosti putem **<fragment>** elementa.
- Na sledećem slajdu prikazane su faze definisanja novog Fragmenta sa postavljanjem željenog izgleda (datoteka šeme).

9.2 - Faze nastajanja fragmenta

Faza I: **Kreiranje fragmenta**

- *onAttach()*
- *onCreate()*
- *onCreateView()*
- *onActivityCreated()*

Faza II: **Fragment je vidljiv**

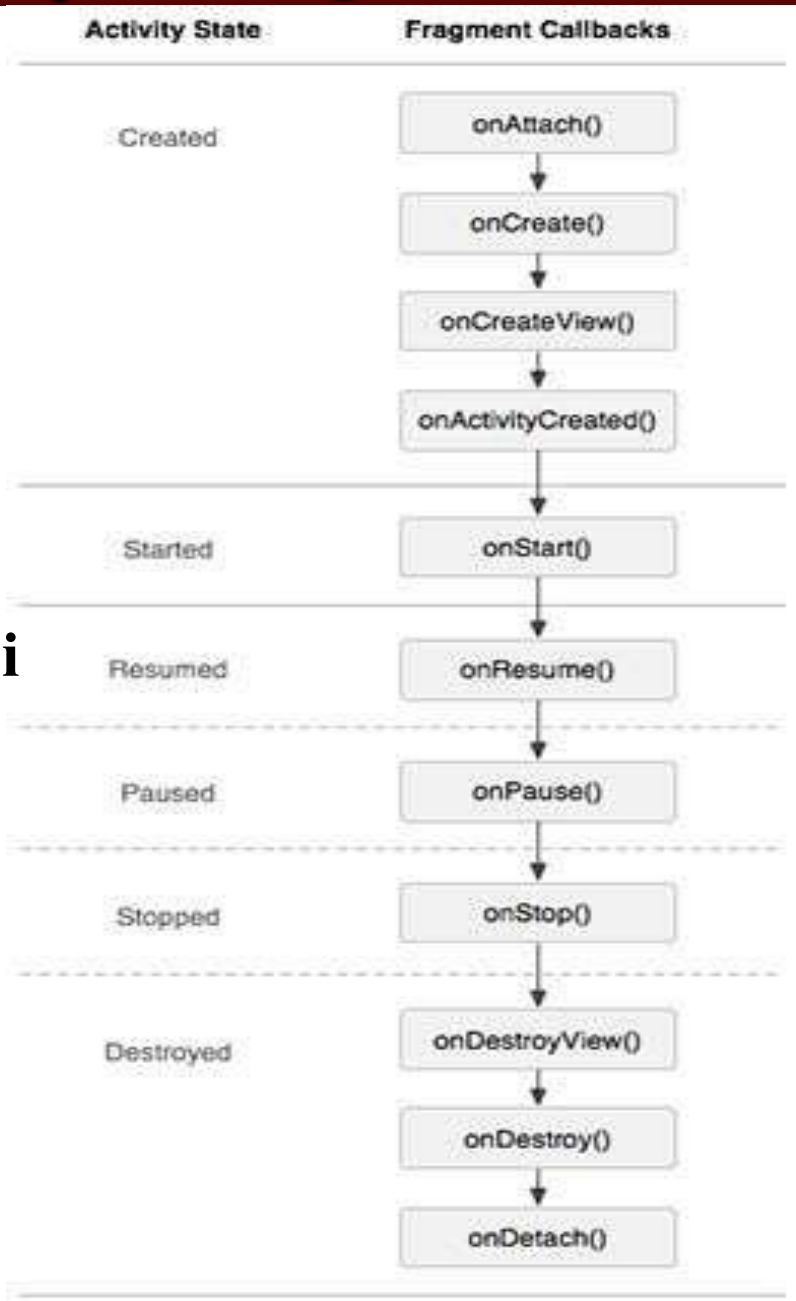
- *onStart()*
- *onResume()*

Faza III: **Fragment se nalazi u pozadini**

- *onPaused()*
- *onStop()*

Faza IV: **Fragment se uništava**

- *onPaused()*
- *onStop()*
- *onDestroyView()*
- *onDestroy()*
- *onDetach()*



9.2 – Kreiranje fragmentata

1. Prvo je potrebno odlučiti koliko fragmenta želimo da ima aktivnost.Na primer, želimo da koristimo dva fragmenta koji će prikazivati sliku u ***landscape*** i ***portrait*** prikazu
2. Na osnovu broja fragmenata potrebno je kreirati klase koje će proširiti ***Fragment*** klasu. ***Fragment*** klasa će sadržati prethodno prikazane ***callback*** funkcije koje možete prilagoditi svojim potrebama.
3. U zavisnosti od funkcije svakog fragmenta potrebno je kreirati posebne XML ***layout*** fajlove koji će imati poseban raspored elemenata
4. Na kraju potrebno je izmeniti datoteku aktivnost za definisanje stvarne logike primene fragmenata u aktivnosti.

9.2 - Definisanje fragmenta

```
public class ExampleFragment : Fragment
{
View rootView;
public override void OnCreate(Bundle savedInstanceState)
{
base.OnCreate(savedInstanceState);
}
public override View OnCreateView(LayoutInflater inflater,
ViewGroup
container, Bundle savedInstanceState)
{
// postavljanje izgleda Fragmenta
rootView = inflater.Inflate(Resource.Layout.example_fragment,
container, false);
// pretraga pogleda, dodeljivanje vrednosti
return rootView;
}
}
```

9.2 - Smeštanje fragmenta u aktivnost

```
public class MainActivity : Activity
{
protected override void OnCreate(Bundle
savedInstanceState)
{
base.OnCreate(savedInstanceState);
SetContentView(Resource.Layout.Main);
// Container – objekat u koji se smesta Fragment
var container =
FindViewById<LinearLayout>(Resource.Id.container);
// kreiranje novog Fragmenta, dodavanje u Activity i na stek
ExampleFragment ef = new ExampleFragment();
FragmentManager.BeginTransaction().Add(container.Id, ef,
"example").
AddToBackStack(null).Commit();
}
}
```

9.2 – Kreiranje fragmentata

- Lista ***callback*** metoda koje treba izmeniti u novo kreiranoj klasi:
  1. **onCreate()** - sistem poziva ovu metodu prilikom kreiranja fragmenta. Ovde treba definisati ključni deo fragmenta koji želimo da zadržimo kada je fragment pauziran ili zaustavljen pa ponovo pokrenut.
  2. **onCreateView()** - sistem poziva ovu metodu kada je potrebno da fragment prikaže korisnički interfejs prvi put. Potrebno je vratiti ***View*** komponentu iz ovog metoda koji je osnov za prikaz fragmenta.
  3. **onPause()** - ovaj metod se poziva kao prvi pokazatelj da korisnik napušta fragment. Ovde treba uneti sve izmene koje je treba da ostanu trajne i van trenutne sesije korisnika.

Primer:

*Na jednostavno primeru aplikacije za prikaz **Hello World**, potrebno je kreirati dva fragmenta od kojih jedan služi za prikaz teksta u **landscape** modu a drugi fragment za prikaz teksta u **portrait** modu.*

9.3 – Intenti

- Tri ključne komponente svake aplikacije – **aktivnosti**, **servisi** i **broadcast risiveri** se aktiviraju preko poruka – ***intents***.
- Razmena poruka preko intenta omogućava kasno povezivanje, tokom izvršavanja, između komponenata u okviru iste ili različitih aplikacija.
- Objekat ***Intent*** je pasivna struktura podataka koja drži apstraktan opis operacija koje treba da se izvrše, ili u slučaju ***broadcast risivera***, opis nečega što se dogodilo ili se objavljuje.
- Android sistem pronalazi odgovarajuću aktivnost, servis ili broadcast risiver kao odgovor na **intent** poruku.
- Android poseduje pravila za mapiranje intenta sa komponentama, odnosno na koji način se određuje komponenta koja prima **intent** poruku.
- Za intente koji nemaju eksplicitno definisanu ciljnu komponentu, neophodno je testirati **Intent** objekte u odnosu na **intent filtere** povezane sa potencijalnim ciljnim komponentama.

9.4 – Intenti

- Postoje odvojeni mehanizmi za dopremanje intenta svakoj od tri navedene komponente:
  1. Intent objekat se šalje ka ***Content.Activity()*** ili ***Activity.startActivityForResult()*** da bi se pokrenula aktivnost ili pozvala postojeća aktivnost da uradi nešto novo. Intent može biti poslat ka ***Activity.setResult()*** da bi se vratila informacija ka aktivnosti koja je pozvala ***startActivityForResult()***.
  2. Intent objekat se šalje ka ***Context.startService()*** da bi se inicirao servis ili dostavile nove instrukcije ka aktivnom servisu. Slično, intent može biti poslat ka ***Context.bindService()*** da bi se uspostavila veza između komponente koja poziva i ciljnog servisa. Opciono može se inicirati servis ako nije aktivan.
  3. Intent objekti koji se šalju nekoj od boradcast metoda kao što su: ***Context.sendBroadcast()***,***Context.sendOrderedBroadcast()***, ili ***Context.sendStickyBroadcast()***) se dostavljaju svim zainteresovanim ***broadcast risiverima***.

9.3 – Intenti

- Intent objekat je paket informacija.
- Sadrži informacije od interesa za komponentu koja prihvata intent (na primer akcija koju treba preduzeti i podaci nad kojima se radi), kao i informacije od interesa za Android sistem (kao na primer kategorija komponenti koja treba da prihvati intent i instrukcije kako da pokerene ciljnu aktivnost).
- Svaki Intent objekat sadrži:

1. ***Ime komponente***
2. ***Action*** – akciju koju treba realizovati
3. ***Data*** – podaci o URI gde se nalaze podaci I MIME tip podataka
4. ***Category*** – podaci o komponenti koja treba da izvrši intent
5. ***Extrats*** - uređeni parovi koji nose dodatnu vrednost za komponentu koja obrađuje Intent
6. ***Flags*** - sadrži instrukcije u slučaju pokretanja aktivnosti

9.4 – Sadržaj Intenta

1. ***Ime komponente***
   - ✓ Sadrži ime komponente koja treba da upravlja Intentom.
   - ✓ Ovo polje je objekat ***ComponentName*** – kombinacija imena klase ciljne komponente (na primer ***android.elab.project.app.ImeAktivnosti***) i ime ***package***-a postavljenog u manifest fajlu aplikacije u kojoj se komponenta nalazi (na primer ***android.elab.project***).
   - ✓ Ime komponente se postavlja pomoću ***setComponent()***, ***setClass()*** ili ***setClassName()***, a čita se pomoću ***getComponent()***.

2. ***Action***
   - ✓ String koji definiše akciju koju treba realizovati, ili u slučaju ***broadcast*** Intenta, akciju koja se desila i o kojoj se izveštava.
   - ✓ Intent klasa definiše veći broj konstanti, obuhvatajući sledeće:

9.4 – Konstante Intent klase

Konstanta	Ciljna komponenta	Akcija
ACTION_CALL	activity	Iniciranje poziva
ACTION_EDIT	activity	Prikaz podataka za editovanje korisniku
ACTION_MAIN	activity	Započni kao inicijalnu aktivnost u okviru aplikacije, bez ulaznih i izlaznih podataka
ACTION_SYNC	activity	Sinronizuje podatke na serveru sa podacima na mobilnom uređaju.
ACTION_BATTERY_LOW	broadcast receiver	Obaveštenje da je baterija pri kraju.
ACTION_HEADSET_PLUG	broadcast receiver	Slušalice su utaknute ili istaknute iz uređaja
ACTION_SCREEN_ON	broadcast receiver	Ekran je uključen
ACTION_TIMEZONE_CHANGED	broadcast receiver	Podešavanja za vremensku zonu su promenjena

9.4 - Sadržaj Intenta - Akcija

- Akcija u velikoj meri definiše kako je struktuiran ostatak intenta, posebno polja ***data*** i ***extras***.
- Akcija u Intent objektu se postavlja pomoću metode ***setAction()*** , a čita se pomoću ***getAction()***.
- Neki od primera uređenog para **action/data** su:

1. **ACTION_VIEW content://contacts/people/1** – prikazuje informacije o korisniku čiji je identifikator "1".
2. **ACTION_VIEW tel:123** - prikazuje telefonski brojčanik i prikazuje navedeni telefonski broj.
3. **ACTION_EDIT content://contacts/people/1** – vrši izmenu podataka o osobi čiji je identifikator "1".
4. **ACTION_VIEW content://contacts/people/** - prikazuje listu osoba kroz koju korisnik može da pretražuje. Selektovanjem osobe za detaljniji prikaz otvara se novi intent.

- Podaci o URI-ju na kome se nalaze podaci i MIME tip podataka.
- Različite akcije su uparene sa različitim tipovima podataka.

9.4 - Sadržaj Intenta - Podaci

- Ukoliko je action polje **ACTION_EDIT** data polje će sadržati URI dokumenta koji je potrebno prikazati za edit-ovanje.
- Ukoliko je action polje **ACTION_CALL** data polje će biti *tel: URI* sa brojem telefona koji je potrebno pozvati.
- Ako je action polje **ACTION_VIEW** a data polje je *http:URI* pozvana aktivnost će ***download***-ti i prikazati ono na šta upućuje navedeni URI.
- Kada se uparuju intenti i komponente koje su sposobne da obrađuju podatke potrebno je znati tip podataka (MIME tip) kao dodatak URI-ju
- Komponentu koja prikazuje slike ne bi trebalo pozivati kada je potrebno pustiti audio fajl.
- Tip podataka se može zaključiti i iz URI-ja naročito iz ***content: URI*** koji ukazuje da su podaci smešteni na uređaju i kontrolisani od strane dobavljača sadržaja.
- Međutim tip podataka se može eksplicitno postaviti u Intent objektu.
- ***setData()*** metod navodi podatke kao URI, ***setType()*** navodi podatke kao MIME tip a ***setDataAndType()*** navodi podatke i kao URI i MIME
- Podatke o URI-ju dobijamo sa ***getData()*** a podatke o tipu sa ***getType()***

9.4 - Sadržaj Intenta - Kategorija

- Kategorija je string koji sadrži dodatne informacije o komponenti koja treba da obradi Intent.
- **Intente klasa** definiše nekoliko predefinisanih kategorija:

Kategorija	Značenje
CATEGORY_BROWSABLE	Ciljana aktivnost se može pozvati iz browser-a za prikaz podataka koji su referencirani linkom (slika ili e-pošta).
CATEGORY_GADGET	Aktivnosi mogu biti ugrađene u druge aktivnosti koje hostuju alate (gadget)
CATEGORY_HOME	Aktivnost prikazuje početni ekran tj. Prvi ekran koji korisnik vidi kada je uređaj uključen i dugme Home je pritisnuto.

- Metoda ***addCategory()*** smešta kategoriju u Intent objekat, ***removeCategory()*** briše prethodno dodatu kategoriju i ***getCategory()*** uzima skup svih kategorija koje su trenutno u objektu.

9.4 - Sadržaj Intenta - Extras i Flags

- Predstavljaju uređene parove koji nose dodatnu vrednost za komponentu koja obrađuje Intent.
- Kao što su neke akcije uparene sa specifičnim vrstama URI-ja tako su i neke akcije uparene sa posebnim dodacima.
- Na primer, **ACTION_TIMEZONE_CHANGED** Intent ima "time-zone" extra koji identifikuje novu vremensku zonu, **ACTION_HEADSET_PLUG** ima "state" extra koji ukazuje na to da li su slušalice uključene ili isključene kao i "name" extra koji definiše tip slušalica.
- Intent objekat ima više ***put...()*** metoda za ubacivanje različitih tipova extra podataka i analogne ***get...()*** methoda za čitanje ovih podataka.

Flags

- Android sistemu je moguće zadati instrukcije u slučaju pokretanja aktivnosti (kojem zadatku pripada aktivnost) i kako je obraditi ukoliko je pokrenuta (da li ju je potrebno dodati u listu prethodnih aktivnosti).
- Svi flegovi se definišu u Intent klasi.

9.5 – Vrste Intenta

- Intente možemo podeliti u dve grupe:
  1. ***Eksplicitne intente*** – određuju ciljnu komponentu po imenu. Korišćenje imena komponenti nije dobar način za korišćenje komponenata iz drugih aplikacija pa se ovaj pristup koristi za interne poruke aplikacije (za pokretanje podređenog servisa ili za pokretanje aktivnosti na istom nivou).
  2. ***Implicitni intenti*** ne imenuju ciljnu komponentu (ime kopmonente je prazno). Implicitni intenti se često koriste za pokretanje aktivnih komponenti u drugim aplikacijama.
- Android isporučuje eksplicitne **intent**-e instanci dizajnirane klase.
- Ništa osim imena komponente nije bitno za određivanje komponente koja obrađuje **Intent**.
- Eksplicitni Intenti se najčešće koriste u okviru jedne aplikacije.
- Ukoliko je dobro poslat, Intent će aktivirati klasu Aktivnost2:

```
Intent  i = new Intent (this, Aktivnost2.class);
```

9.5 – Vrste Intenta

- Implicitni Intenti ne specificiraju Java klasu koju treba pozvati.
- Oni definišu naziv akcije koju treba realizovati i opciono URI koji treba upotrebiti za akciju.

Primer: *sledeći intent govori Android OS-u da prikaže web stranicu. Iako je web brauzer registrovan za ovaj intent i druge komponente mogu biti prijavljene za ovaj intent*:

```
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.myelab.net"));
```

- Kada se implicitni intent pošalje Androidu, on pretražuje sve komponente registrovane za određenu akciju i tip podataka.
- Ako Android pronađe samo jednu komponentu, on je direktno startuje.
- Međutim, ako Android identifikuje više komponenata registrovanih za taj intent, otvara se dijalog koji traži od korisnika da definiše komponentu pomoću koje će biti obrađen intent.
- Komponenta koja prihvata intent može preuzeti informacije iz intenta preko metoda **getAction()** i **getData()**.

9.5 – Vrste Intenta

- Komponenta koja kreira intent može dodati podatke na intent preko metode **putExtra()**.
- ***Extras*** su parovi ključ/vrednost; a ključ je uvek ***String***.
- Kao vrednost uzima bilo koji primitivni tip podataka:*String*,*Bundle,* sl.
- Na primer, sve komponente koje su registrovane za slanje podataka mogu se aktivirati preko ***new Intent(Intent.ACTION_SEND)***.

```
Intent sharingIntent = new Intent(Intent.ACTION_SEND);
sharingIntent.setType("text/plain");
sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT,
"News for you!");
// createChooser is a convenience method to create
// an Chooser Intent with a Title
startActivity(Intent.createChooser(sharingIntent,"Share this
using"));
```

9.5 – Vrste Intenta

- Komponenta koja prihvata intent, može uzeti podatke pomoću *getIntent().getExtras()*

```
Bundle extras = getIntent().getExtras();
if (extras == null) {
                return;
                }
// Get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
     // Do something with the data
}
```

9.6 – Uloga Intent filtera

- Za impicitne **Intent**-e Android mora pronaći najbolju komponentu za obradu Intent-a – jednu aktivnost, servis da izvrši traženu radnju ili skup broadcast prijemnika kao odgovor na broadcast objavu.
- To se vrši upoređivanjem sadržaja Intent objekta i ***intent filters***-a – strukture povezane sa komponentama.
- **Filteri** oglašavaju mogućnosti komponente i razdvajaju Intent-e na one koji mogu i koji ne mogu da obrade zahtev i objašnjavaju koje implicitne intent-e komponente mogu da obrade
- Ako komponenta nema intent filter ona može da obrađuje samo eksplicitni intent.
- Komponente sa filterima obrađuju i eksplicitne i implicitne intent-e
- Tri aspekta Intent objekta se posmatraju kada se poredi sa intent filter:
  1. **Action**
  2. **Data** (i **URI** i **tip podataka**)
  3. **Category**
- **Extras** i **flag**-ovi nemaju nikakvog uticaja na rešavanje problema koja komponenta može da obradi Intent

9.6 – Uloga Intent filtera

- Zadatak je da informišu sistem koje implicitne filtere mogu da obrade.
- **Aktivnosti**, **servisi** i **broadcast prijemnici** mogu da imaju jedan ili više Intent filtera.
- Svaki filter opisuje mogućnosti komponente odnosno set Intenta koji je komponenta spremna da obradi.
- Na ovaj način filteri blokiraju neželjene Intente ali samo neželjene implicitne intente.
- Eksplicitni intenti se uvek isporučuju bez obzira na sadržaj, odnosno filter se u ovim slučajevima uopšte ne posmatra.
- Impicitni intenti se isporučuju komponenti samo ukoliko prođu jedan filter komponente.
- Komponente imaju različite filtere za svaki posao koji može da obavi.

Primer: *NoteEditor aktivnost Note Pad aplikacije ima dva filtera – jedan za pokretanje određene beleške koju korisnik može pregledati ili izmeniti i drugi za pokretanje nove, prazne beleške koju korisnik može popuniti i sačuvati.*

9.6 – Sigurnost Intent filtera

- Oslanjanje na filtere kada je sigurnost u pitanju nije dovoljna.
- Iako otvara komponentu samo za određene zahteve od impicitnih filtera ne čini ništa da zaštiti komponente od eksplicitnih intenta.
- Intent filter je instanca ***IntentFilter*** klase.
- Kako Android OS mora znati mogućnosti komponente pre nego što je pokrene, intent filteri se ne postavljaju u Java kodu nego u manifest fajlu aplikacije (**AndroidManifest.xml**) kao ***<intent-filter>*** element.
- Filter ima polja koja su zadužena za *action*, *data* i *category* polja Intent
- Implicitni intent se testira filterima iz sve tri oblasti.
- Da bi se isporučio komponenti on mora proći sva tri testa, odnosno filtere iz sve tri oblasti.
- Ukoliko ne prođe **bar jedan** filter Android sistem ga neće isporučiti komponenti.
- Međutim ukoliko komponenta ima više intent filtera u svakoj oblasti dovoljno je da zadovolji bar jedan filter iz oblasti kako bi ga sistem isporučio komponenti.
- U narednim slajdovima biće objašnjeni svaki od ova tri testa detaljnije.

9.6 – Action test

- Element **<intent-filter>** u manifest fajlu prikazuje listu ***action*** elemenata u formi **<action>** taga:

```
<intent-filter . . . >
   <action android:name="com.example.project.SHOW_CURRENT" />
   <action android:name="com.example.project.SHOW_RECENT" />
   <action android:name="com.example.project.SHOW_PENDING" />
   . . .
</intent-filter>
```

- Lista ne može biti prazna, filter mora sadržati barem jedan **<action>** element, u suprotnom će blokirati sve intente.
- Da bi prošao ovaj test **action** definisan u intent objektu se mora poklapati sa barem jednom od akcija definisanih u filteru.
- Ako objekat ili filter ne navodi nijednu akciju, ishodi su sledeći:

1. Ukoliko filter nema navedenu nijednu akciju ne postoji ništa sa čime intent može da se podudara tj. svi intenti ne prolaze test tj. nijedan intent ne prolazi kroz filter.
2. Ukoliko Intent objekat ne precizira akciju automatski prolazi test sve dok filter sadrži barem jednu akciju

9.6 – Category test

- Element **<intent-filter>** prikazuje listu kategorija kao podelemente.

```
<intent-filter . . . >
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    . . .
</intent-filter>
```

- Konstante definisane u manifest fajlu se ne koriste već se koriste puni nazivi stringa.
- Na primer "**android.intent.category.BROWSABLE**" string je primer koji korespondira sa **CATEGORY_BROWSABLE** konstantom definisanom ranije.
- Slično i string "**android.intent.action.EDIT**" korespondira sa **ACTION_EDIT** konstantom.
- Da bi intent prošao test kategorije svaka kategorija u intent objektu mora se poklapati sa kategorijom u filteru.
- Filter može imati i dodatne kategorije ali mora sadržati sve koje su u intent-u.

9.6 - Category test

- Iz tog razloga Intent objekat bez kategorija uvek prolazi test bez obzira šta je u filteru.
- Međutim postoji jedan izuzetak:
- Android sve implicitne intent-e koji su prosleđeni ka **startActivity()** metodi tretira kao da sadrže barem jednu kategoriju i to "**android.intent.category.DEFAULT**" (**CATEGORY_DEFAULT** konstantu).
- Iz tog razloga aktivnosti koje žele da primaju implicitne intente moraju uključiti "**android.intent.category.DEFAULT**" u svoje intent filtere.
- Filteri sa "**android.intent.action.MAIN**" i "**android.intent.category.LAUNCHER**" podešavanjima su izuzeci.
- Oni označavaju aktivnosti koje počinju nove zadatke i koji se prikazuju na početnom ekranu.
- One mogu uključivati "**android.intent.category.DEFAULT**" u listi kategorija ali ne moraju.

9.6 – Data test

- Kao i **action** i **category** i **data** specifikacija je u vidu podelementa i može se pojavljivati više puta. Na primer:

```
<intent-filter . . . >
   <data android:mimeType="video/mpeg" android:scheme="http" . . . />
   <data android:mimeType="audio/mpeg" android:scheme="http" . . . />
   . . .
</intent-filter>
```

- Svaki **<data>** element može definisati **URI** i tip podataka (**MIME** tip)
- Postoje posebni atributi: **scheme**, **host**, **port** i **path** za svaki deo URI-ja: **scheme://host:port/path**

- Na primer kod sledećeg URI-a

  content://com.example.project:200/folder/subfolder/etc

  scheme je "***content***“
  host je "***com.example.project***“
  port je "***200***“
  path je "***folder/subfolder/etc***".

9.6 – Data test

- Host i port zajedno čine URI ***authority***; ako host nije definisan port se ignoriše.
- Svaki od ovih atributa je opcion ali nisu nezavisni među sobom.
- ***Authority*** zavisi od ***scheme*** dok ***path*** to zavisi i od ***scheme*** i od ***authority***-*a*.
- Kada se URI u intent objektu poredi sa specifikacijom URI-ja u filteru on se poredi samo sa definisanim delovima u filteru.
- Ukoliko filter definiše samo ***scheme***, svi URI-ji sa tom šemom se poklapaju sa filterom.
- Samo ***path*** specifikacija u filteru može sadržati specijalne znakove pomoću kojih se omogućuje delimično poklapanja ***path***-a.

9.6 – Intent filteri

- Atribut **type** u **<data>** elementu definiše MIME tip podataka i mnogo se češće koristi nego URI filter.
- Intent objekat i filter mogu koristiti "*" kao specijalni karakter koji zamenjuje sve podtipove tog polja.
- Primer: "**text/***" ili "**audio/***" - dozvoljavaju poklapanje i sa svim podtipovima.
- Data test poredi i URI i tip podataka u Intent objektu sa URI-jem i tipom podataka definisanom u filteru.
- Važe sledeća pravila:

1. Intent objekat koje ne sadrži ni URI ni tip podataka prolazi test samo ukoliko filter ne specificira nikakve URI ni tipove podataka.
2. Intent objekat koji sadrži URI a ne sadrži nikakav tip podataka (i tip podataka se ne može zaključiti iz URI-ja) prolazi test samo ako se URI poklapa sa URI-jem u filteru i filter ne specificira nikakve tipove podataka.
3. Intent objekat koji sadrži tipove podataka ali ne sadrži URI prolazi test samo ukoliko filter definiše isti tip podataka a ne specificira URI.

9.6 – Intent filteri

4. Intent objekat koji sadrži i URI i tip podataka (ili se tip podataka može zaključiti iz URI-ja) prolazi test samo ukoliko se tip podataka poklapa se tipom u filteru. URI test ili ako njegov URI odgovara URI-ju u filteru ili ukoliko ima **content:** ili **file: URI** a filter ne specificira URI. Za komponentu se porazumeva da podržava **content:** i **file:.**

- Poslednje pravilo odražava očekivanje da su komponente sposobne da izvuku lokalne podatke iz fajla ili od strane **content provajdera**.
- Iz tog razloga njihovi filteri mogu navesti samo listu tipova podataka a ne moraju eksplicitno navoditi imena **content:** i **file:** šema.
- Element **<data>** kao u primeru govori Androidu da komponenta može da uzme sliku od content dobavljača i da je prilaže:

```
<data android:mimeType="image/*" />
```

9.6 – Intent filteri

- S obzirom da se većina podataka dobavlja od strane content dobavljača filteri koji određuju tip podataka a ne definišu URI su najčešći.
- Filteri koji definišu ***scheme*** i **tip podataka** su takođe veoma česti.
- Na primeru je prikazan **<data>** element koji govori Androidu da komponenta može da dopremi video materijal sa mreže i da ga prikaže:

```
<data android:scheme="http" android:type="video/*" />
```

- Ukoliko Web pretraživač sledi link ka nekoj Web stranici on prvo pokušava da prikaže sadržaj te stranice i uspeva ukoliko je reč o HTML stranici.
- Međutim ukoliko ne može da prikaže podatke on sastavlja implicitni intent sa ***scheme*** i **tipom podataka** i pokreće aktivnost koja može da obradi zahtev.
- Ukoliko nema nikoga da obradi zahtev on aktivira preuzimača datoteka kako bi preuzeo podatke.
- Nakon toga ga preuzima ***content provider*** odnosno mnogo više aplikacija može da odgovori na taj zahtev.

9.6 – Intent filteri

- Sledeća specifikacija prijavljuje aktivnost za intent koji se poziva kada neka komponenta zatraži otvaranje Web stranice:

```
<activity android:name=".BrowserActivitiy"
android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http"/>
  </intent-filter>
</activity>
```

- Primer definisanja aktivnosti za intent koji traži ACTION_SEND za text/plain tip podataka

```
<activity
          android:name=".ActivityTest"
   android:label="@string/app_name" >
   <intent-filter>
     <action android:name="android.intent.action.SEND" />
     <category android:name="android.intent.category.DEFAULT" />
     <data android:mimeType="text/plain" />
   </intent-filter>
</activity>
```

9.6 – Intent filteri

- Mnoge aplikacije započinju bez referenciranja na podatke.
- Aktivnosti koje inicijalizuju aplikacije imaju filtere "**android.intent.action.MAIN**" koji su specificirani kao akcija.
- Ako želimo da budu zastupljeni u ***application launcher***-u potrebno je da definišemo i kategoriju "**android.intent.category.LAUNCHER**":

```
<intent-filter . . . >
    <action android:name="code android.intent.action.MAIN" />
    <category android:name="code
android.intent.category.LAUNCHER" />
</intent-filter>
```

9.7 – Korišćenje poklapanja Intent-a

- Intenti se uparuju sa intent filterima ne samo da bi se odredilo koju komponentu je potrebno aktivirati nego i iz razloga otkrivanja nečega o setu komponenti instaliranih na uređaju.

Primer: *Android sistem popunjava application launcher, tj. početni ekran koji prikazuje sve aplikacije koje su dostupne korisniku za pokretanje pronalaženjem svih aktivnosti čiji intent filter specificira* ***action***

"**android.intent.action.MAIN**" i
"**android.intent.category.LAUNCHER**"

kategoriju (kao u ranije navedenom primeru).

- On nakon toga prikazuje sve ikonice i labele tih aktivnosti u ***application launcher***-u.
- Na isti način pronalazi home ekran traženjem svih aktivnosti koje imaju "**android.intent.category.HOME**" u svom filteru.

9.7 - Korišćenje poklapanja Intent-a

- Svaka aplikacija može veoma jednostavno koristiti uparivanje intent-a.
- ***PackageManager*** klasa ima skup **query...()** metoda koje vraćaju sve komponente koje mogu da prihvate određeni intent i skup sličnih **resolve...()** metoda koje određuju komponentu koja najbolje odgovara zadatom intent-u.

Primeri:

1. ***queryIntentActivities()*** vraća listu svih aktivnosti koje mogu da obrade intent koji je prosleđen kao parametar
2. ***queryIntentServices()*** vraća listu svih servisa koji mogu da obrade intent koji je prosleđen kao parametar dok metod,
3. ***queryBroadcastReceivers()*** vraća listu ***broadcast receiver***-a.

- Ove metode ne aktiviraju komponente nego samo vrše izlistavanje onih komponeneti koje mogu da obrade tražene zahteve.

Hvala na pažnji !!!

Pitanja

? ? ?

9.3 – Primer

1. Kreiraćemo u Android Studiju aplikaciju pod imenom ***MyFragments*** u okviru paketa ***com.example.myfragments***, sa ***blank Activity*** opcijom.
2. Potrebno je mdifikovati fajl *MainActivity.java*. U okviru njega potrebno je ispitati orijentaciju prikaza (***Landscape*** ili ***Portrait***) i shodno tome izabrati odgovarajuće fragmente.
3. Kreiraćemo dva Java fajla ***PM_Fragment.java*** i ***LM_Fragement.java*** u okviru paketa ***com.example.myfragments*** kako bi definisali fragmente i odgovarajuće metode u njima.
4. Za oba fragmenta definišimo i layout-e ***res/layout/lm_fragment.xml*** i ***res/layout/pm_fragment.xml***
5. U okviru fajla ***res/layout/activity_main.xml*** treba izmeniti kod kako bi uključili oba novo definisana fragmenta.
6. Unesite vrednosti korišćenih konstanti u fajl ***res/values/strings.xml***.
7. Pokrenite aplikciju u okviru Android emulatora i proverite da li unete promene pravilno prikazuju ***Landscape*** ili ***Portrait*** prikaze.

9.3 – MainActivity.java fajl

Sadržaj **MainActivity.java** fajla koji se nalazi u **src/com.example.mycontentprovider/MainActivity.java**:

```
package com.example.myfragments;
import android.os.Bundle;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;
import android.view.WindowManager;
public class MainActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
Configuration config = getResources().getConfiguration();
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
/* Check the device orientation and act accordingly */
if (config.orientation == Configuration.ORIENTATION_LANDSCAPE)
{
/* Landscape mode of the device */
LM_Fragment ls_fragment = new LM_Fragment();
fragmentTransaction.replace(android.R.id.content, ls_fragment);
}else{
/* Portrait mode of the device */
PM_Fragment pm_fragment = new PM_Fragment();
fragmentTransaction.replace(android.R.id.content, pm_fragment);
}
fragmentTransaction.commit();
}
}
```

9.3 - Fajl LM_Fragment.java za prikaz fragmenta

- Kreiraćemo dva fajla

1. **LM_Fragement.java**
2. **PM_Fragment.java**

koji će sadržati progr. kod za prikaz fragmenata

- Njih treba smestiti u folder ***com.example.mycontentprovider.***
- Sa desne strane prikazan je programski kod fajla: **LM_Fragement.java**

```
package com.example.myfragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class LM_Fragment extends
Fragment{
@Override
public View onCreateView(LayoutInflater
inflater,
ViewGroup container, Bundle
savedInstanceState) {
/**
* Inflate the layout for this fragment
*/
return inflater.inflate(
R.layout.lm_fragment, container, false);
}
}
```

9.3 - Program.kod PM_Fragment.java

```
package com.example.myfragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class PM_Fragment extends Fragment{
@Override
public View onCreateView(LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
/**
* Inflate the layout for this fragment
*/
return inflater.inflate(
R.layout.pm_fragment, container, false);
}
}
```

9.3 - Layout fajlovi za fragmente

- Potrebno je kreirati i dva layout fajla **lm_fragement.xml** i **pm_fragment.xml**
- Fajlove treba smestiti u direktorijumu ***res/layout***.
- Sa desne strane prikazan je sadržaj fajla **lm_fragement.xml**

```
<?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"
android:background="#7bae16">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/landscape_message"
android:textColor="#000000"
android:textSize="20px" />
<!-- More GUI components go here -->
</LinearLayout>
```

9.3 - Layout fajl pm_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#666666">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/portrait_message"
android:textColor="#000000"
android:textSize="20px" />
<!-- More GUI components go here -->
</LinearLayout>
```

9.3 - Fajl activity_main.xml

- Fajl se nalazi u direktorijumu **res/layout/activity_main.xml**
- On uključuje dva fragmenta:
  1. **lm_fragment**
  2. **pm_fragment**

  koja smo prethodno formirali

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal">
<fragment
android:name="com.example.fragments"
android:id="@+id/lm_fragment"
android:layout_weight="1"
android:layout_width="0dp"
android:layout_height="match_parent" />
<fragment
android:name="com.example.fragments"
android:id="@+id/pm_fragment"
android:layout_weight="2"
android:layout_width="0dp"
android:layout_height="match_parent" />
</LinearLayout>
```

9.3 - Sadržaj fajla strings.xml

- Fajl treba smestiti u direktorijumu **res/values/strings.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">MyFragments</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
<string name="landscape_message">This is
Landscape mode fragment
</string>
<string name="portrait_message">This is Portrait
mode fragment
</string>
```

9.3 – Pokretanje aplikacije

- Da bi promenili način prikaza na ekranu našeg emulatora (ili telefona) potrebno je pritisnuti **control+F11** čime biramo *landscape* u *portrait* prikaz.
- Prilikom promene prikaza aktiviraće se različiti GUI koje smo ranije formirali za *landscape* u *portrait* prikaze.
- U okviru iste aktivnosti imamo različite prikaze zahvaljujući različitim fragmentima
- To znači da mi možemo da koristimo različite GUI komponente za različite prikaze prema zahtevima aplikacije u okviru samo jedne aktivnosti