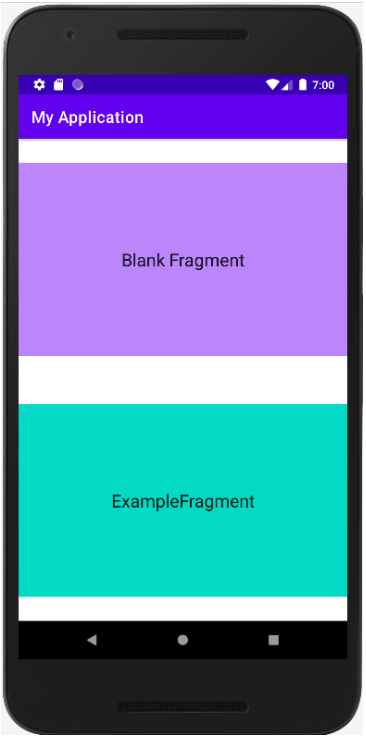
# Android Fragmenti

## Uvod u Android Fragmente

**Fragment** je samostalni modularni deo korisničkog interfejsa aplikacije koji može da bude ugrađen kao deo Aktivnosti. Fragment može da se ugradi u jednu ili više Aktivnosti. Takođe, jedna Aktivnost može da sadrži više od jednog Fragmenta. Samim tim, korišćenje Fragmenta pruža mogućnost ponovne upotrebe elemenata. U toku izvršenja aplikacije, mogu se izvršavati različite transakcije nad Fragmentima (dodavanje Fragmenta u Aktivnost, uklanjanje, zamena...).

Slično kao Aktivnosti, i Fragmenti su funkcionalni elementi sa sopstvenim životnim ciklusom. Međutim, Fragmenti se upotrebljavaju samo kao deo Aktivnosti i ne mogu da postoje kao samostalni elementi aplikacije!

Slika 1 prikazuje izgled Aktivnosti koja sadrži dva Fragmenta.




*Slika 1 - Aktivnost koja sadrži dva Fragmenta*

## Kreiranje Fragmenta

Fragment čine dve komponente:

- XML opis rasporeda elemenata u Fragmentu
- Java klasa koja rukuje Fragmentom i nasleđuje klasu **Fragment**.

Novi Fragment se u Android Studio dodaje na sledeći način: **File -> New -> Fragment -> Fragment (Blank)**. Moguće je dodati različite tipove Fragmenta. Ovim koracima se u aplikaciju dodaje layout za XML opis Fragmenta kao i Java klasa koja rukuje Fragmentom.

Primer kreiranja XML opisa Fragmenta koji se sastoji iz LinearLayout elementa plave pozadine i TextView elementa unutar njega:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/blue">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center_vertical"
      android:gravity="center"
      android:text="@string/hello_fragment"/>
</LinearLayout>
```

## Dodavanje Fragmenta u Aktivnost

Postoje dva načina dodavanja Fragmenta u Aktivnost. Prvi način je ubacivanjem Fragmenta u XML opis Aktivnosti, a drugi način je dodavanje Fragmenta pomoću Java koda.

Fragment se u XML opis Aktivnosti dodaje XML elementom **<fragment>**. Ključni atributi unutar elementa <fragment> su **android:name** i **tools:layout**. **Android:name** atribut povezuje Fragment sa Java klasom koja ga opisuje, dok **tools:layout** atribut ukazuje na XML opis Fragmenta koji se dodaje u Aktivnost. Obavezno je dodeliti i **id** atribut, a pozicija Fragmenta u Aktivnosti se definiše na isti način kao i za bilo koji drugi View ili ViewGroup element.

Primer XML rasporeda Aktivnosti sa dodatim Fragmentom:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
```

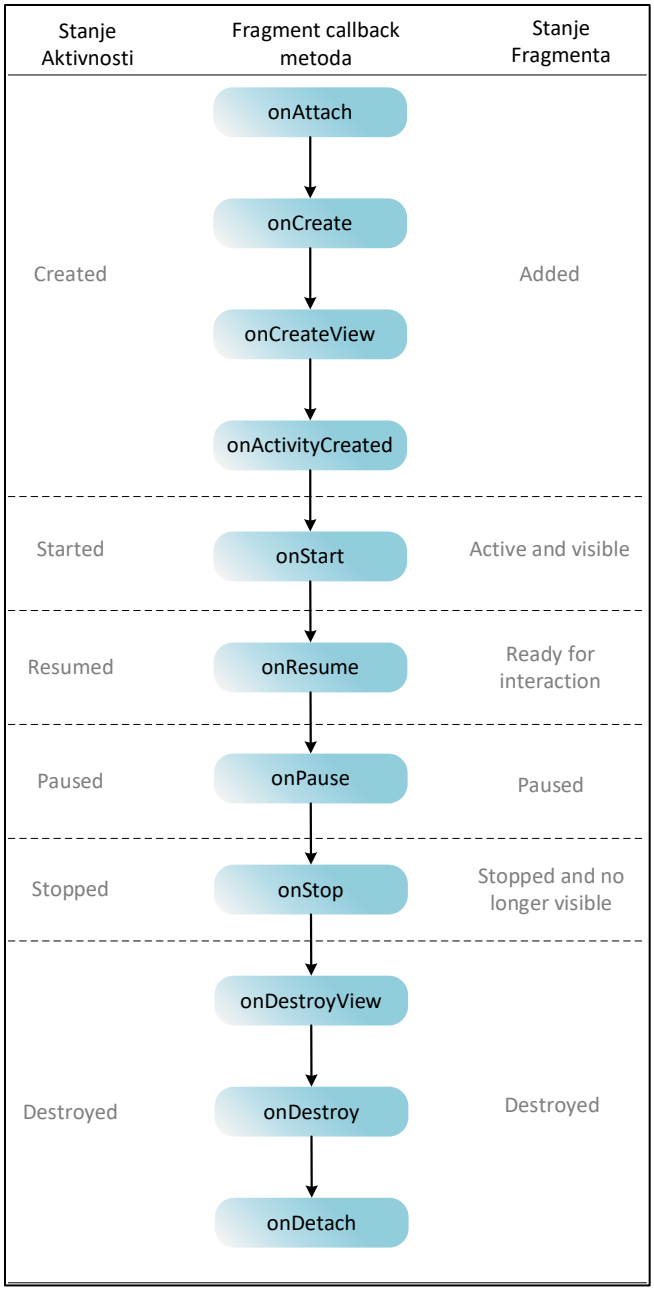
```
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_height="match_parent"
        tools:context=".MainActivity"
        android:gravity="center">

        <fragment
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:name="com.example.fragmentexample.ExampleFragment"
            tools:layout="@layout/fragment_example"
            android:layout_gravity="center"
            android:id="@+id/fragment1"/>
    </LinearLayout>
```

Ovakav način dodavanja Fragmenta ima za manu to što Aktivnost ne može da ukloni Fragment u toku izvršenja. Kako bi se postigla potpuna dinamička kontrola Fragmenta u toku izvršenja aplikacije, Fragmente treba dodati u Aktivnosti pomoću Java koda, što će biti opisano u narednim poglavljima.

# Životni ciklus Fragmenta

Metode životnog ciklusa Fragmenta su slične metodama životnog ciklusa Aktivnosti, s tim da sadrže i dodatne metode koje omogućavaju interakciju sa Aktivnosti koja sadrži Fragment. Kako životni vek Fragmenta direktno zavisi od životnog veka Aktivnosti prikazuje Slika 2.




*Slika 2 - Životni ciklus Fragmenta*

Dodatne metode životnog ciklusa Fragmenta:

- *onAttach* - poziva se kada se Fragment povezuje sa Aktivnošću; u ovoj metodi je korisno proveriti da li je u Aktivnosti implementiran osluškivač Fragmenta
- *onCreateView* - poziva se radi prikazivanja korisničkog interfejsa Fragmenta
- *onActivityCreated* - poziva se kada je Aktivnost kreirana (Aktivnost je prešla u stanje Created); ova metoda se poziva nakon metode onCreateView
- *onDestroyView* - poziva se kada korisnički interfejs Fragmenta više nije vidljiv
- *onDetach* - poziva se kada Fragment više nije povezan sa Aktivnošću.

U metodi *onCreateView* se Java klasa Fragmenta povezuje sa XML rasporedom Fragmenta pomoću LayoutInflater klase:

```
View v = inflater.inflate(R.layout.example_f, container, false);
```

Metoda *inflate* kao povratnu vrednost vraća objekat klase View pomoću koga mogu da se dobavljaju elementi Fragmenta:

```
TextView t = v.findViewById(R.id.tekst);
```

# Povezivanje Fragmenta sa Aktivnošću

Kao što je napomenuto, kako bi se postigla potpuna dinamička kontrola Fragmenta u toku izvršenja aplikacije, potrebno je dodati Fragment u Aktivnost Java kodom. Kako bi se Fragment dodao u Aktivnost, potrebno je ispratiti sledeće korake:

1. U XML rasporedu Aktivnosti postaviti ViewGroup kontejner za fragment (obično je to prazan layout koji zauzima određeno mesto u Aktivnosti, umesto koga će biti postavljen fragment)
2. Kreirati instancu klase koja opisuje Fragment
3. Kreirati instancu transakcije pozvanjem metode *beginTransaction* nad FragmentManager instancom (transakcije omogućuju operacije nad Fragmentima kao što su dodavanje, brisanje, zamena i slično)
4. Pozvati metodu *add* ili *replace* nad instancom trasakcije kako bi se Fragment dodao u postavljeni kontejner; ukoliko se Fragment zameni metodom *replace*, potrebno je pozvati metodu *addToBackStack* kako bi se transakcija postavila u stek (radi eventualnog vraćanja prethodnog Fragmenta pritiskom na *back* dugme)
5. Pozivati metodu *commit* nad transakcijom kako bi se postavile promene.

XML opis rasporeda Aktivnosti sa postavljenim kontejnerom za Fragment:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:orientation="vertical"
   android:layout_height="match_parent"
   tools:context=".MainActivity"
   android:gravity="center">
```

```
    <!-- 1. Postavljanje kontejner elementa -->
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/kontejnerFragmenta"/>
</LinearLayout>
```

Dodavanje Fragmenta iz Java koda Aktivnosti:

```
// 2. Kreiranje instance Fragment klase
ExampleFragment fragment = new ExampleFragment();
// 3. Kreiranje transakcije
// 4. Dodavanje fragmenta
// 5. Postavljanje promena pozivom commit metode
getSupportFragmentManager().beginTransaction()
    .add(R.id.kontejnerFragmenta, fragment)
    .commit();
```

Moguće je izvršiti sledeće operacije (transakcije) nad Fragmentom:

- **add** - dodavanje Fragmenta u Aktivnost
- **remove** - uklanjanje Fragmenta iz Aktivnosti
- **replace** - zamena jednog Fragmenta drugim
- **hide** - skrivanje prikazanog Fragmenta
- **show** - pokazivanje prikazanog Fragmenta
- **detach** - odvajanje Fragmenta od grafičkog interfejsa
- **attach** - spajanje Fragmenta nakon što je odvojen od grafičkog interfejsa.

Kada se pozove transakcija **remove** Fragment se uništava (pozivaju se metode *onPause*, *onStop*, *onDestroyView*, *onDestroy* i *onDetach*). Nakon što je Fragment uništen, on se ponovo može dodati pozivom transakcije **add** (pozivaju se metode *onAttach*, *onCreate*, *onCreateView*, *onStart* i *onResume*).

Sa druge strane, kada se pozove transakcija **detach**, Fragment se ne uništava, samo se odvaja od grafičkog interfejsa (pozivaju se metode *onPause*, *onStop* i *onDestroyView*). Nakon odvajanja, Fragment je moguće ponovo dodati, pozivom transakcije **attach** (pozivaju se metode *onCreateView*, *onStart* i *onResume*).

Primer zamene Fragmenta novim:

```
getSupportFragmentManager().beginTransaction()
    .replace(R.id.kontejnerFragmenta, fragment2)
    .addToBackStack(null)
    .commit();
```

# Obrada događaja unutar Fragmenta

Fragmenti, kao i Aktivnosti, u sebi mogu da sadrže View objekte nad kojima je moguća obrada događaja (Button, ListView…). Primer obrade pritiska na dugme:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup c,
Bundle savedInstanceState) {
    // Inflate the layout for this fragment
   View view =  inflater.inflate(R.layout.fragment_new, c, false);
   Button b = view.findViewById(R.id.button);
   b.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         // handle button click
      }
   });
   return view;
}
```

Takođe, moguće je implementirati *View.OnClickListener* interfejs u Fragment klasi i implementirati *onClick* metodu u kojoj će biti obrađeni svi događaji.

# Komunikacija između Aktivnosti i Fragmenta

Kada je više od jednog Fragmenta ugrađeno u Aktivnost postoji mogućnost da će biti potreban neki vid prenosa podataka između njih (ili između Aktivnosti i Fragmenta). Komunikacija između Fragmenata se obavlja preko posredničke Aktivnosti.

Transakcije nad Fragmentima (add, replace…) kao poslednji (opcioni) parameter omogućuju definisanje jedinstvenog *taga* Fragmenta na osnovu koga Aktivnost može da komunicira sa Fragmentom.

Kako bi se iz Aktivnosti pozvala javna metoda klase Fragment ili pristupilo javnom polju klase potrebno je pronaći Fragment na osnovu njegovog taga:

```
     ExampleFragment f =
(ExampleFragment)getSupportFragmentManager().findFragmentByTag("f");
     f.changeText("Novi tekst");
```