Fragmenti i permisije

Mobilne aplikacije

Pregled sadržaja

Fragmenti

- Fragmenti predstavljaju deo ponašanja ili GUI-a aktivnosti (mogu se posmatrati kao podaktivnosti).
- Jedna aktivnost može da sadrži više fragmenata i jedan fragment može da bude sadržan u više aktivnosti (ali ne ista instanca fragmenta).
- Fragmenti imaju životni ciklus (koji zavisi od životnog ciklusa aktivnosti u kojoj se nalaze) i mogu da obrađuju događaje koje stvara GUI.
- U toku izvršavanja aplikacije se mogu izvršavati transakcije nad fragmentima (mogu se dodavati, uklanjati, zamenjivati, itd.).

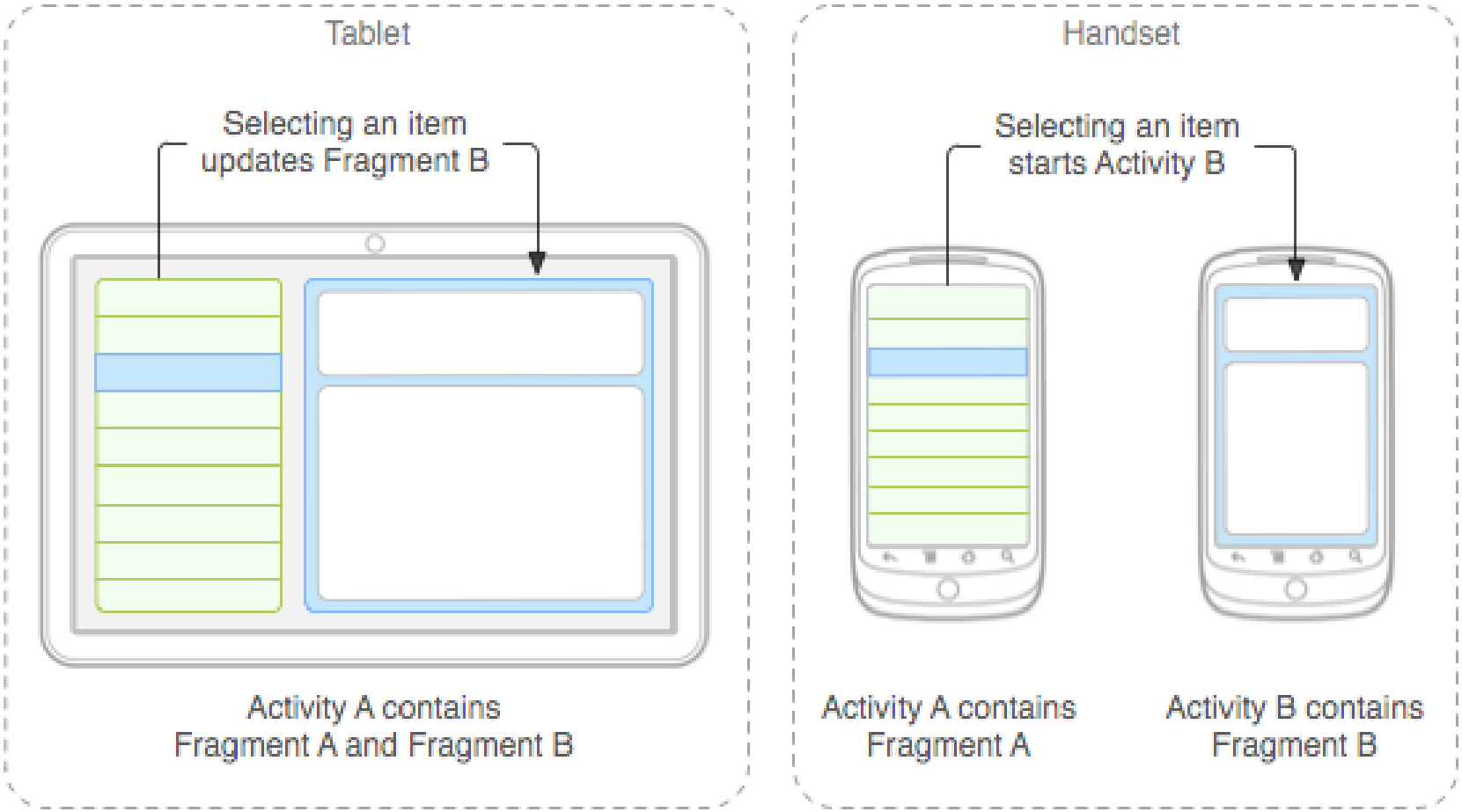


Figure 1: Fragmenti.

Pravljenje fragmenta

- Napisati klasu koja nasleđuje `Fragment` klasu
- Dodati `fragment` element u XML datoteku koja deklariše korisnički interfejs.

Pravljenje fragmenta

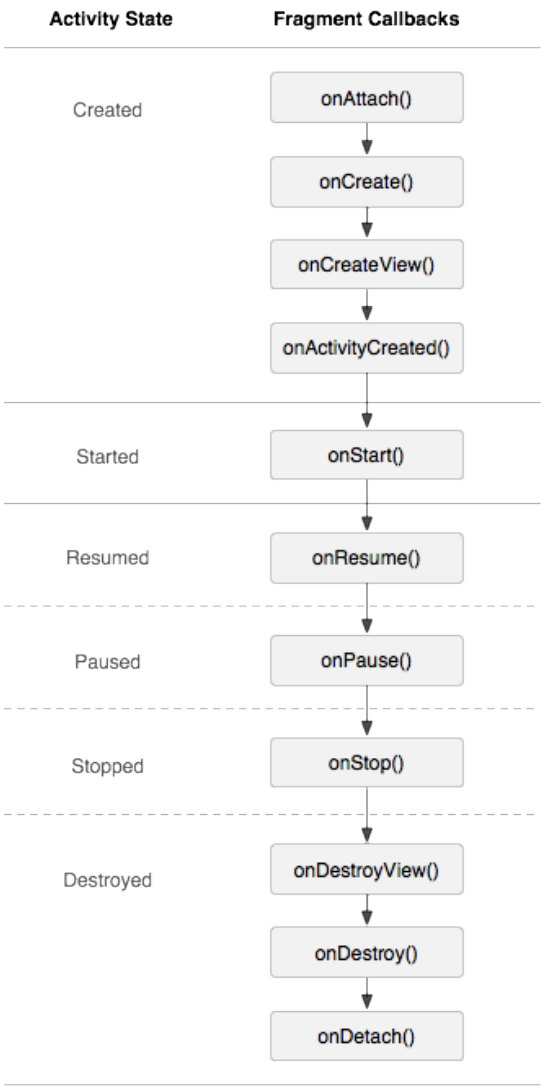
```
package com.example.project;
import android.app.Fragment;

public static class ExampleFragment extends Fragment {
  @Override
  public View onCreateView(...) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.example_fragment, container,
    false);
  }
}
```

Stvaranje fragmenta

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ... >
  <fragment android:name="com.example.project.ExampleFragment" ... />
</LinearLayout>
```

Životni ciklus fragmenta



Životni ciklus fragmenta

Životni ciklus fragmenta je sličan životnom ciklusu aktivnosti, ali oni sadrže dodatne metode koji omogućavaju interakciju sa aktivnošću koja ih sadrži:

- `onAttach` (poziva se kada se fragment povezuje sa aktivnošću)
- `onCreateView` (poziva se da bi se iscrtao korisnički interfejs fragmenta)
- `onActivityCreated` (poziva se kada se onCreate metoda aktivnosti izvrši)
- `onDestroyView` (poziva se da bi se uništio korisnički interfejs fragmenta)
- `onDetach` (poziva se kada se fragment odvezuje od aktivnosti)

Transakcije nad fragmentima

```
// Create new fragment and transaction
ExampleFragment fragment = new ExampleFragment();
FragmentTransaction transaction = getSupportFragmentManager().beginTransaction
    ();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, fragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();

```

Transakcije nad fragmentima

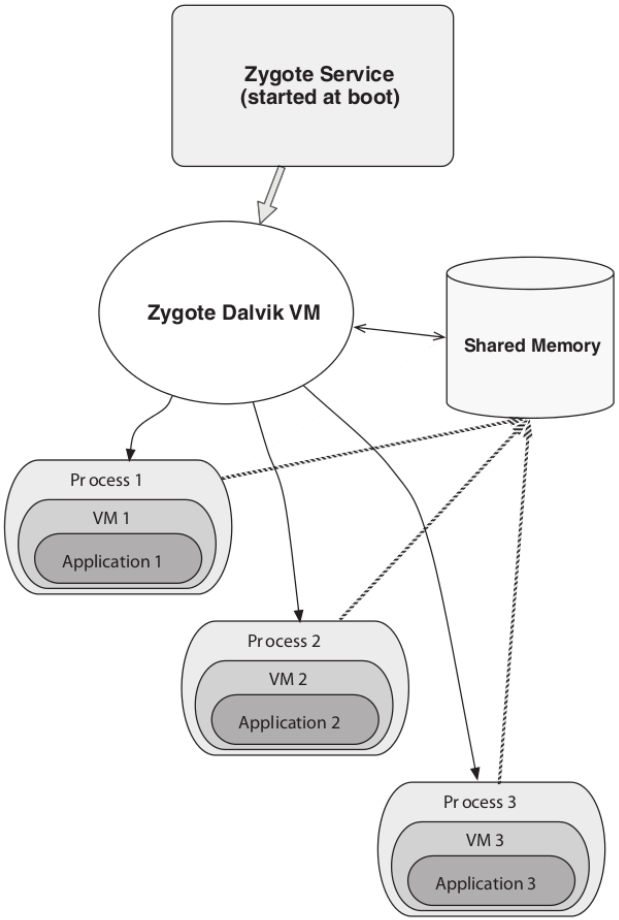
U okviru transakcije je moguće izvršiti sledeće operacije:

- `add` (dodavanje fragmenta u aktivnost)
- `remove` (uklanjanje fragmenta iz aktivnosti)
- `replace` (zamena jednog fragmenta drugim fragmentom)
- `hide` (skrivanje prikazanog fragmenta)
- `show` (prikazivanje skrivenog fragmenta)
- `detach` (odvajanje fragmenta od GUI)
- `attach` (spajanje fragmenta nakon što je odvojen od GUI)

Pregled sadržaja

Prava pristupa

Operativni sistem izoluje aplikacije (kako aplikacije međusobno tako i operativni sistem od aplikacija).



Prava pristupa

- Dodatne funkcije bezbednosti su implementirane mehanizmom prava pristupa
- Aplikacija ne može da izvrši ni jednu operaciju koja može da negativno utiče na druge aplikacije, operativni sistem ili korisnike ukoliko joj to nije dozvoljeno
- Implementacija mehanizma prava pristupa razlikuje se do Androida 5.1 (statička prava pristupa) i od Androida 6.0 (dinamička prava pristupa)

Prava pristupa

Constant	Meaning
`CALL_PHONE`	Allows an application to initiate a phone call.
`SEND_SMS`	Allows an application to send SMS messages.
`RECORD_AUDIOS`	Allows an application to record audio.
`CAMERA`	Required to be able to access the camera device.
`VIBRATE`	Allows access to the vibrator.

Table 1: Prava pristupa.

Prava pristupa

Constant	Meaning
`ACCESS_COARSE_LOCATION`	Allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi.
`ACCESS_FINE_LOCATION`	Allows an app to access precise location from location sources such as GPS, cell towers, and Wi-Fi.
`INTERNET`	Allows applications to open network sockets.
`BLUETOOTH`	Allows applications to connect to paired bluetooth devices.

Table 2: Prava pristupa.

Statička prava pristupa

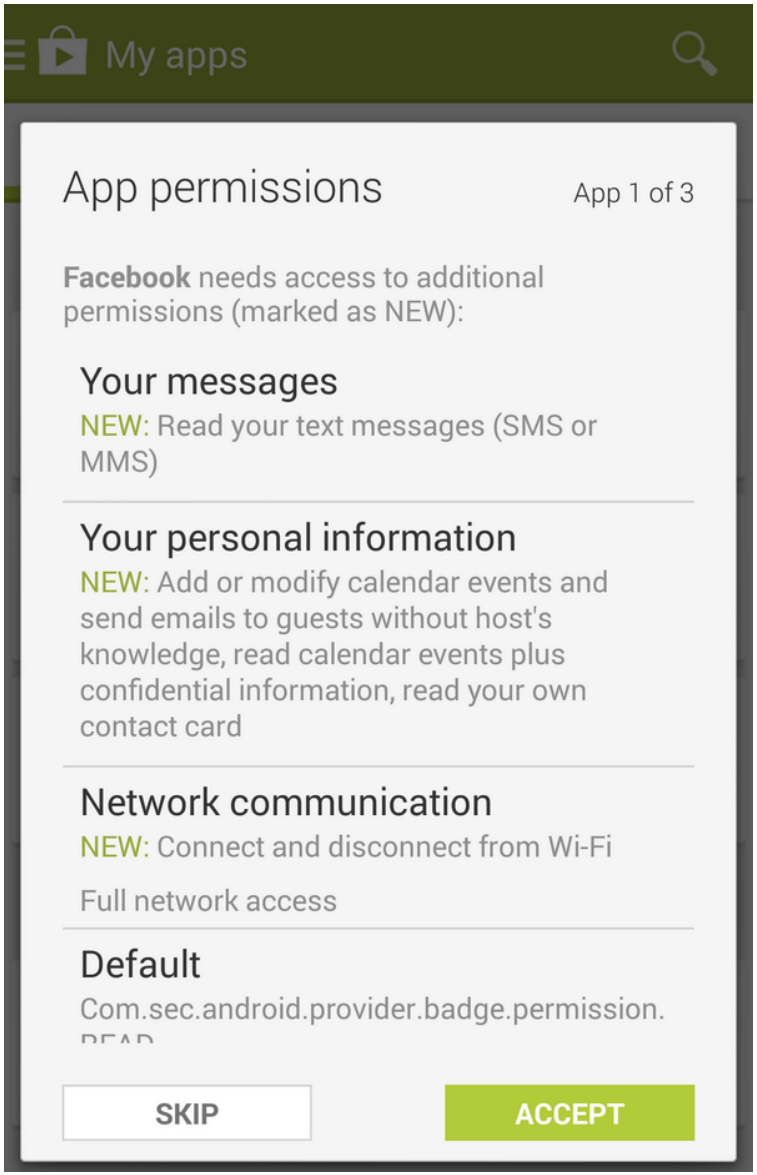
- Do Androida 5.1 prava pristupa koja su potrebna za izvršavanje aplikacije statički se deklarišu u AndroidManifest.xml.
- Korisnik može da aplikaciji prilikom instalacije dodeli prava pristupa koja traži ili da odustane od instalacije aplikacije
- Svaki pokušaj da aplikacija izvrši nedozvoljene operacije biće sprečen

Prava pristupa

```
<manifest ... >
  ...
  <uses-permission android:name="android.permission.INTERNET" />
  ...
</manifest>

```

Statička prava pristupa



Dinamička prava pristupa

- Od Androida 6.0 aplikacija dinamički traži prava pristupa koja su joj potrebna
- To znači da aplikacija mora da svaki put pre nego što izvrši operaciju koja zahteva pravo pristupa proveri da li ima to pravo pristupa
- Android može automatski odobriti aplikaciji pravo pristupa ili može zatražiti od korisnika da joj odobri pravo pristupa (u zavisnosti od osetljivosti operacije i resursa)
- Korisnik ima mogućnost da aplikaciji u svakom trenutku oduzme pravo pristupa

Dinamička prava pristupa

```
// Assume thisActivity is the current activity
int permissionCheck = ContextCompat.checkSelfPermission(
    this,  Manifest.permission.WRITE_CALENDAR);

```

Permisije i nivoi rizika

- U zavisnosti od toga kojim podacima i akcijama ograničavaju pristup, razlikuju se permisije niskog nivoa rizika i permisije visiokog nivoa rizika.
- Permisije niskog nivoa rizika se automatski dozvoljavaju. Dovoljno ih je zatražiti u Manifest fajlu.
- Neke od permisija niskog nivoa rizika su INTERNET, BLUETOOTH, VIBRATE
- Permisije visokog nivoa rizika se moraju dinamički zatražiti tj. korisnik ih mora eksplicitno dozvoliti.
- Primeri permisija visokog nivoa su ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, CALL_PHONE, SEND_SMS, ...

Dinamička prava pristupa

```
private static final int MY_PERMISSIONS_REQUEST_READ_CONTACTS = 0;

...

if (ContextCompat.checkSelfPermission(this, Manifest.permission.
    READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(
          this,
          new String[]{Manifest.permission.READ_CONTACTS},
          MY_PERMISSIONS_REQUEST_READ_CONTACTS);

        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
        // app-defined int constant. The callback method gets both,
    the
        // result of the request paired and this constant.
}
```

Dinamička prava pristupa

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[],
    int[] grantResults) {

  switch (requestCode) {
    case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
      // If request is cancelled, the result arrays are empty.
      if (grantResults.length > 0 && grantResults[0] == PackageManager.
    PERMISSION_GRANTED) {

        // permission was granted, yay! Do the
        // contacts-related task you need to do.

        } else {

          // permission denied, boo! Disable the
          // functionality that depends on this permission.

        }

        return;
     }

    // other 'case' lines to check for other
    // permissions this app might request

    }
}
```