

ANDROID ALKALMAZÁSFEJLESZTÉS

Android komponensek használata

Activity

Fragment

Service

Broadcast Receiver

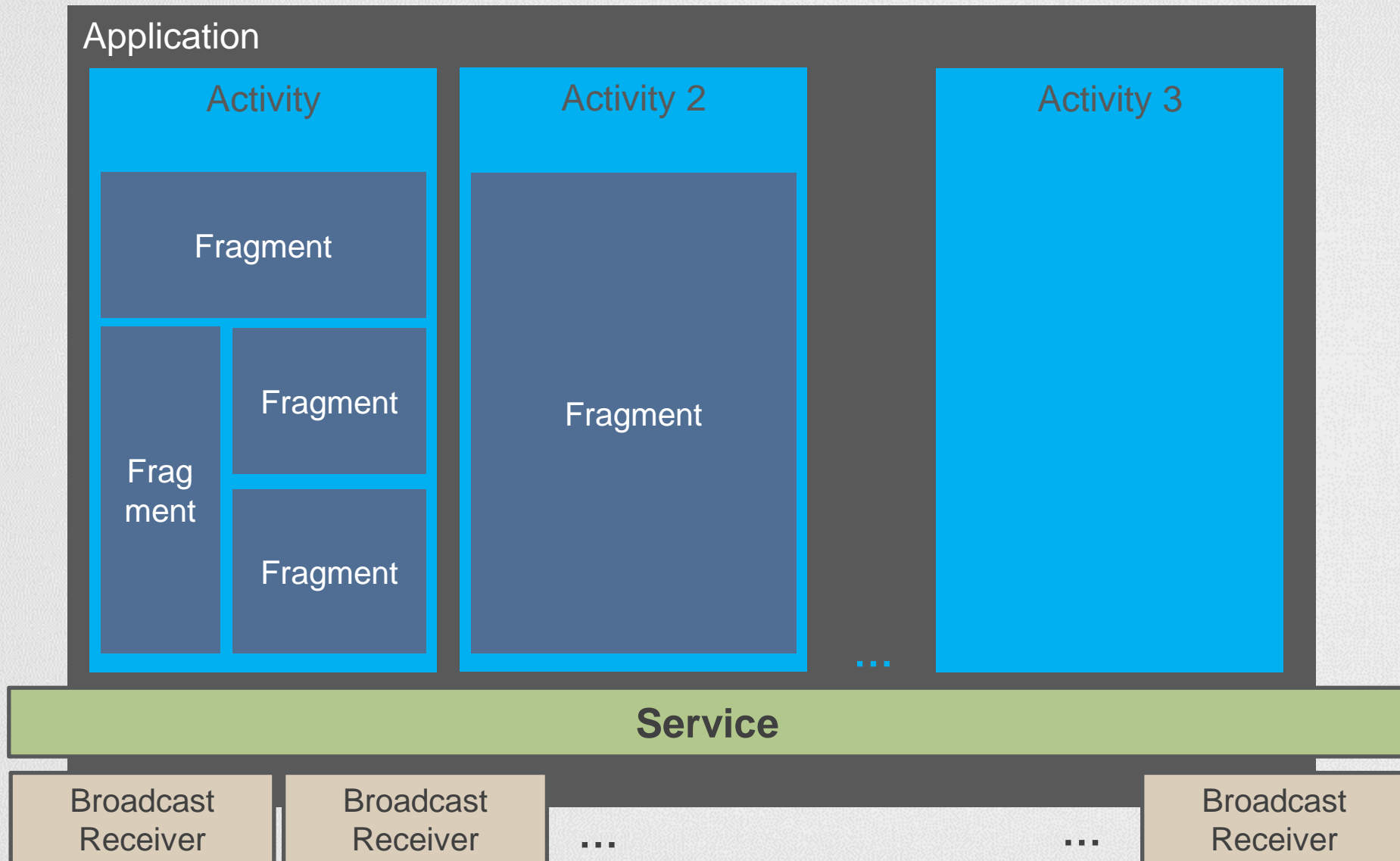


sicz-mesziar.janos@nik.uni-obuda.hu

Sicz-Mesziár János

2013. szeptember 27.

Overview



Application

Az alkalmazást írja le, mely az alábbi főbb komponenseket tartalmazhatja:

- Activity
- Service
- Broadcast Receiver

Fontosabb paraméterek:

- Icon – az alkalmazás ikonja, ez jelenik meg a Launcher-en is
- Label – az alkalmazás neve
- HardwareAccelerated – hardveres gyorsítás engedélyezése
- Theme – az alkalmazásra kiterjedő stílus
- Large heap – alkalmazás által használható maximális memóriaméret ideiglenes megemlése ha szükséges
- [további paraméterek...](#)

Activity

- Leírja, hogy hogyan működjön egy felhasználói felület, azaz ez tölti be a kontroller szerepét.
- Saját [életciklusa](#) van.
- Fontosabb működést befolyásoló paraméterek:
 - `icon` – az adott activity ikonja, például a „recent apps”-ban ez látszik
 - `label` – adott activity megnevezése
 - `hardwareAccelerated` – hardveres gyorsítás engedélyezése activity-n belül
 - `screenOrientation` – portrai, landscape, reverseLandscape, sensor, ...
 - `configChanges` – fejlesztő kezel bizonyos változásokat: orientation, screenSize
 - `launchMode` – activity példány működés
 - `windowSoftInputMode` – billentyűzet beállítások
 - `theme` – Activity-re alkalmazott stílus fájl, például `noWindowTitle`
 - `exported` – Activity elindítható third-party applikáció által is

Új Activity indítása

Új Activity hozzáadása:

`[package] > jobb klikk > New > Activity`

Új Activity hozzáadása kézzel:

- Layout XML (`res/layout/uj_activity.xml`)
- Activity leszármoztatása (`src/[package] /UjActivity.java`)
- AndroidManifest.xml-ben felvenni

Új Activity indítása

```
Intent masikActivity = new Intent(this, Masik.class);
startActivity(masikActivity);
```

Paraméterek átadása

Primitívek átadása Activity-nek

```
Intent masikActivity = new Intent(this, Masik.class);
masikActivity.putExtra("szam", 100);
masikActivity.putExtra("key", "érték");
startActivity(masikActivity);
```

Primitívek

integer, double, float, string, boolean, byte, short, char

Átadott paraméterek elérése

```
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    int szam = getIntent().getExtras().getInt("szam", 0);
    String kulcs = getIntent().getExtras().getString("key", null);
}
```


Paraméterek átadása (2) – Parcelable

```
public class MyModel implements Parcelable {
    private String name;
    private int value;
    public MyModel(String name, int value) { ... }

    public MyModel(Parcel parcel){
        this.name = parcel.readString();
        this.value = parcel.readInt();
    }
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeString(name);
        parcel.writeInt(value);
    }

    public static final Creator<MyModel> CREATOR = new Creator<MyModel>() {
        public MyModel createFromParcel(Parcel parcel) {
            return new MyModel(parcel);
        }
        ...
    };
}
```

Összetett objektum
átadása paraméterként.

Paraméterek átadása (2) – Visszatérés

Activity indítása visszatérési értékre várva

```
Context.startActivityForResult(Intent i, int requestCode);
```

Visszatérési érték meghatározása

```
setResult(RESULT_OK, new Intent());
```

Visszatérési érték levétele

```
public class MainActivity extends Activity {  
    protected void onActivityResult(int requestCode,  
                                    int resultCode,  
                                    Intent data) {  
        data.getIntExtra("kulcs", -1);  
        super.onActivityResult(requestCode, resultCode, data);  
    }  
}
```


Service

Egy alkalmazás komponens:

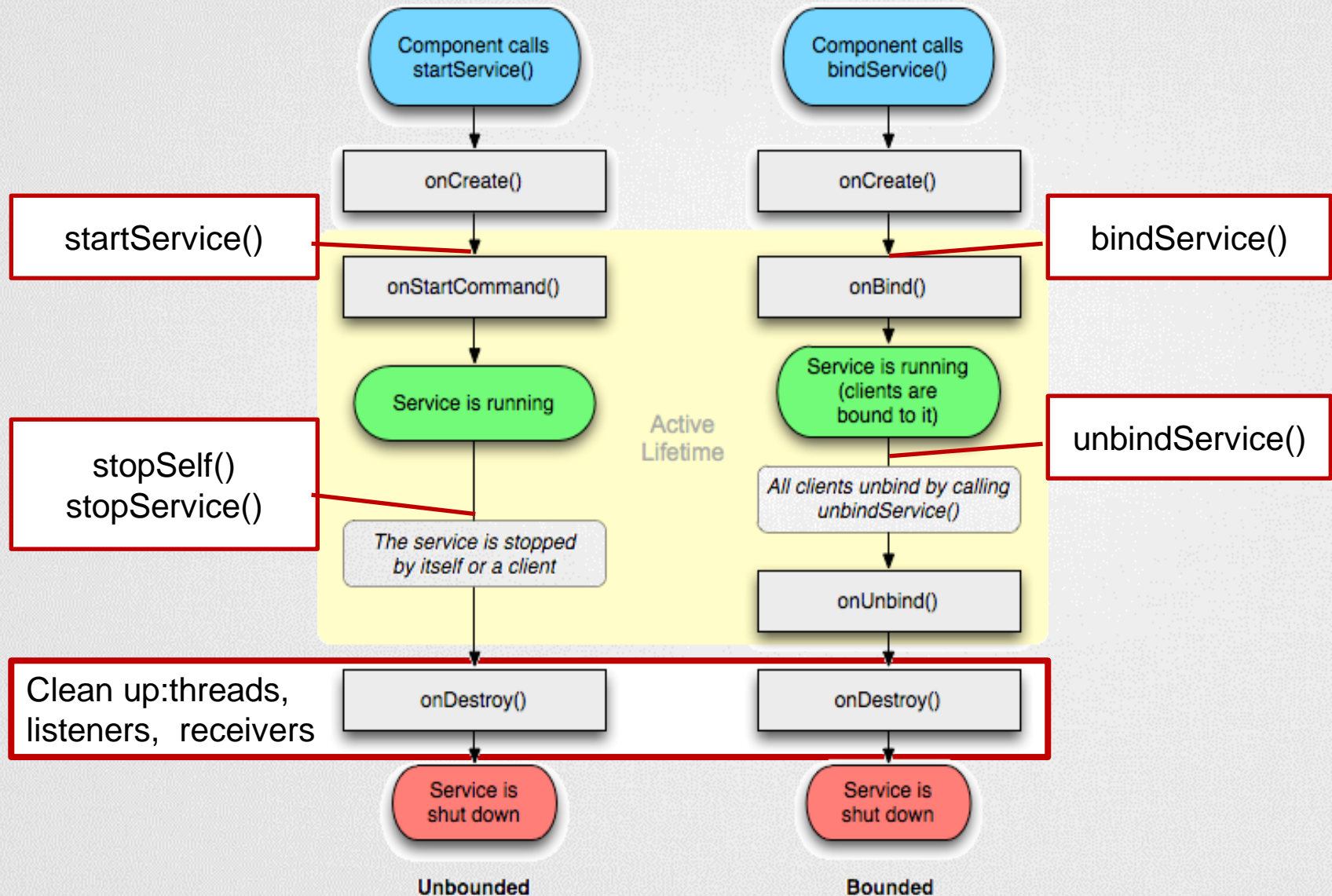
- hosszan futó műveletek végrehajtása
- háttérben fut, nincs UI

Két formája van:

- Started
 - egy komponens (pl.: Activity) elindítja a `startService()` eljárással
 - Végtelenségig fut, akkor is ha az őt elindító komponens megsemmisül. (Nincs visszatérési érték) → **stopService()**
 - Általában egy műveletet hajt végre, majd megsemmisíti magát. Pl.: letöltés / feltöltés az internet irányába
- Bound
 - Egy komponens `bindService()`-al kötődik
 - Kliens-szerver felület a komponens és service között (request, results, ...)
 - Addig fut amíg az összeköttetés él. (Pl.: zenelejátszás)

**Nem külön szálon fut!
Nem egy külön folyamat!**

Service (2)



Broadcast Receiver

Aszinkron, üzenetszórásos értesítés, azok akik felregisztráltak rá értesülnek róla

Két fő csoport:

- Normal broadcast
 - teljesen aszinkron,
 - „véletlen” sorrend
 - Hatékonyabb, de korlátok: nincs visszatérési érték, visszavonás
- Ordered broadcast
 - egyszerre csak egy „receiver” fut
 - visszavonható – abort
 - Prioritásokat adhatunk (android:priority)

onReceive() csak egyszer fut le - rövid életű

- Hosszan futó műveletekre alkalmatlan (timeout ~10sec)

Broadcast Receiver a gyakorlatban

Regisztrálás statikusan

- [BroadcastReceiver](#) osztály implementálása
- AndroidManifest.xml-ben <receiver> megadása

Regisztrálás dinamikusan, futási időben

- BroadcastReceiver osztály implementálása
- Context.registerReceiver()-el regisztrálunk → onResume()
- Context.unregisterReceiver() leiratkozunk → onPause()

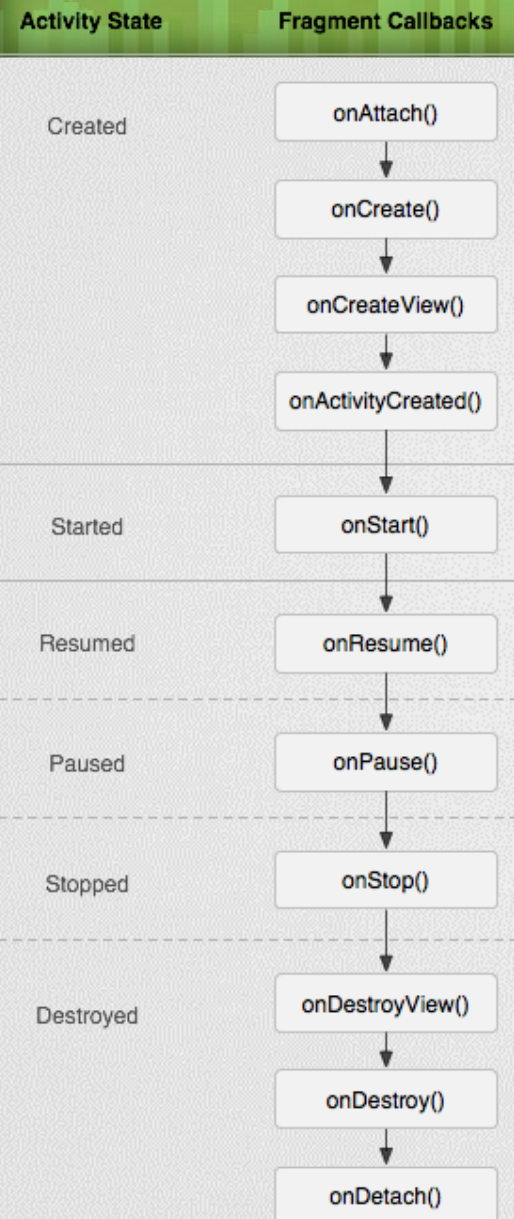
```
private BroadcastReceiver mBatInfoReceiver = new  
BroadcastReceiver(){  
    public void onReceive(Context context, Intent intent){  
        int level = intent.getIntExtra("level",0);  
        contentTxt.setText("" + level + "%");  
    }  
};
```

Broadcast kibocsátás

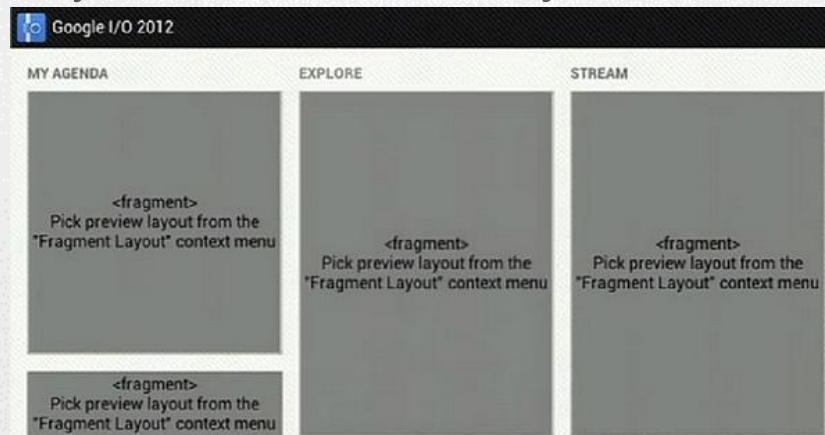
- Context.sendBroadcast(Intent intent)

Fragments





- Activity moduláris része, saját életrajzzal:
 - Ha egy Activity „*pause*” állapotba kerül akkor az összes benne lévő Fragment is!
 - Függetlenül manipulálható: futás időben hozzáadható, elvehető, „back stack”-be tehető.
 - Layout-on bárhol elhelyezhető.



Lásd
[Google IO 2012](#)
[App demo](#)

- Android 3.0 (API level 11) óta érhető el
- Android 1.6-tól is használható Support Library segítségével.

Fragment a gyakorlatban

```
public class MyListFragment extends Fragment {  
    public View onCreateView(  
        LayoutInflater inflater,  
        ViewGroup container,  
        Bundle savedInstanceState) {  
  
        View v = inflater.inflate(  
            R.layout.list_fragment,  
            container,  
            false);  
  
        return v;  
    }  
}
```

↑
Fragment kinézetét leíró XML

Fragment betöltése

Fragment betöltése futás időben

```
private void loadFragment(){
    FragmentManager fm = getSupportFragmentManager();
    FragmentTransaction ft = fm.beginTransaction();
    ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
    ft.addToBackStack(MyListFragment.class.getName());
    ft.replace(R.id.container_place, new MyListFragment());
    // ft.add(R.id.container_place, new MyListFragment());
    // ft.remove(Fragment...);
    // ft.hide(Fragment...);
    ft.commit();
}
```

Backstack

- Hasonló az alkalmazás stack-hez
- Fragment váltások előzményeit tartalmazza
- Vissza gomb (back) esetén automatikusan betölti az előző fragmentet
- Ennek állapota lekérdezhető:
 - `getBackStackEntryAt(int index)`
 - `getBackStackEntryCount()`
 - `popBackStack()`
 - `addOnBackStackChangeListener(OnBackStackChangeListener listener)`