

Mobil alkalmazásfejlesztés - Üzenetszórás, Szolgáltatások

Dr. Bilicki Vilmos

Szoftverfejlesztés Tanszék

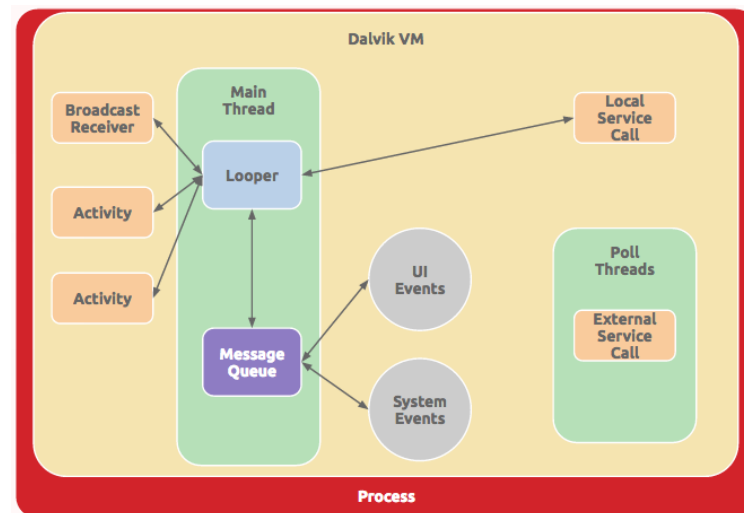
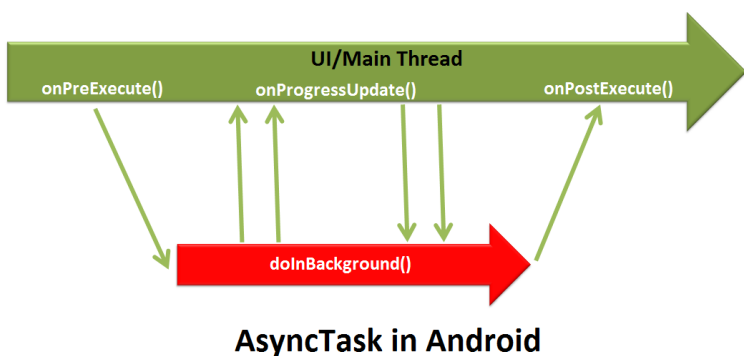
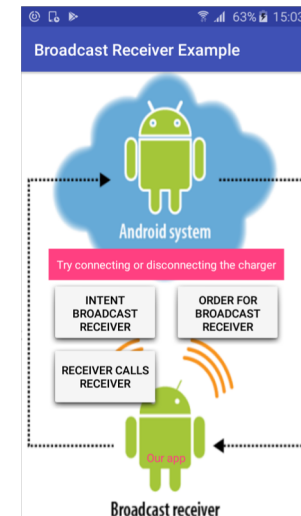


A fóliához felhasznált anyagok:

Google: Android Developer Fundamentals (Version 2), <https://developer.android.com/courses/fundamentals-training/overview-v2>

Az előző előadás

- ▶ Háttér folyamatok
 - AsyncTask és AsyncTaskLoader
 - Internet kapcsolat



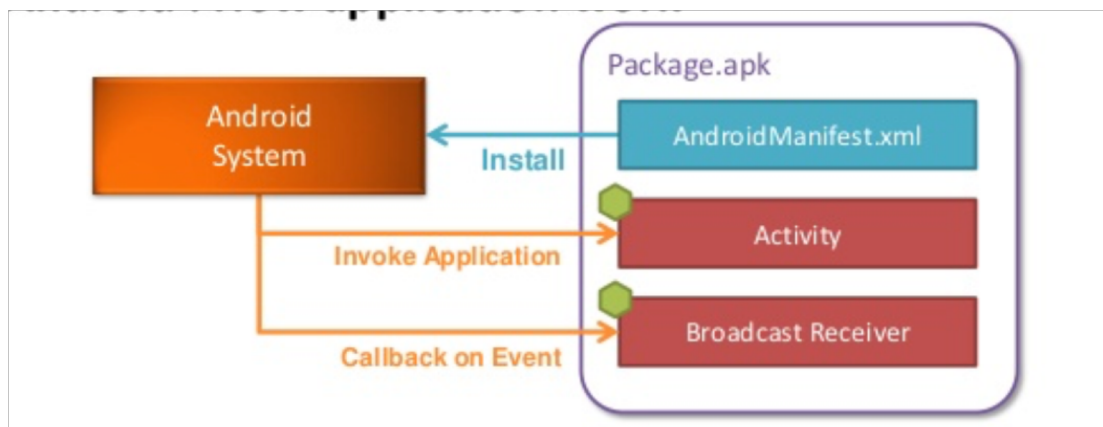
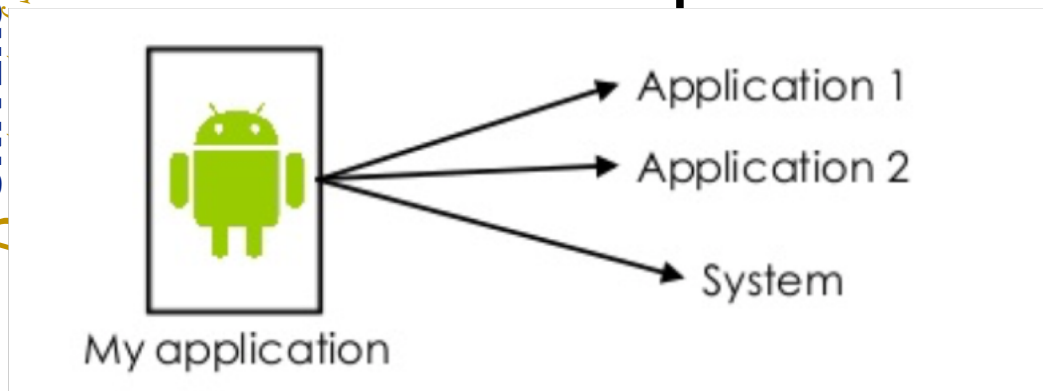
A mai előadás tartalma

- ▶ Üzenetszórás
 - Áttekintés
 - Egyedi üzenetszórás
 - Üzenetszórás vevők
 - Üzenetszórás vevők megvalósítása
 - Üzenetszórás szűrése
 - Legjobb gyakorlat
- ▶ Szolgáltatások



Emlékeztető

- ▶ Intentek segítségével küldhetünk üzeneteket a processzus határokon túl







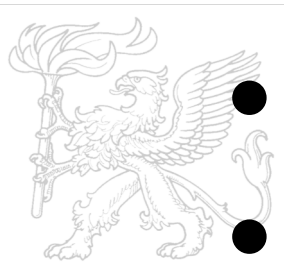
Üzenetszórás

Események hatására az Android rendszer és az alkalmazások üzenetszórás üzeneteket küldhetnek

Az üzenetek `Intent` objektumokba vannak burkolva.

Ez az `Intent` objektum tartalmazza az esemény részleteit, pl.: `android.intent.action.HEADSET_PLUG`, akkor küldjük amikor a fejhallgató állapota változik

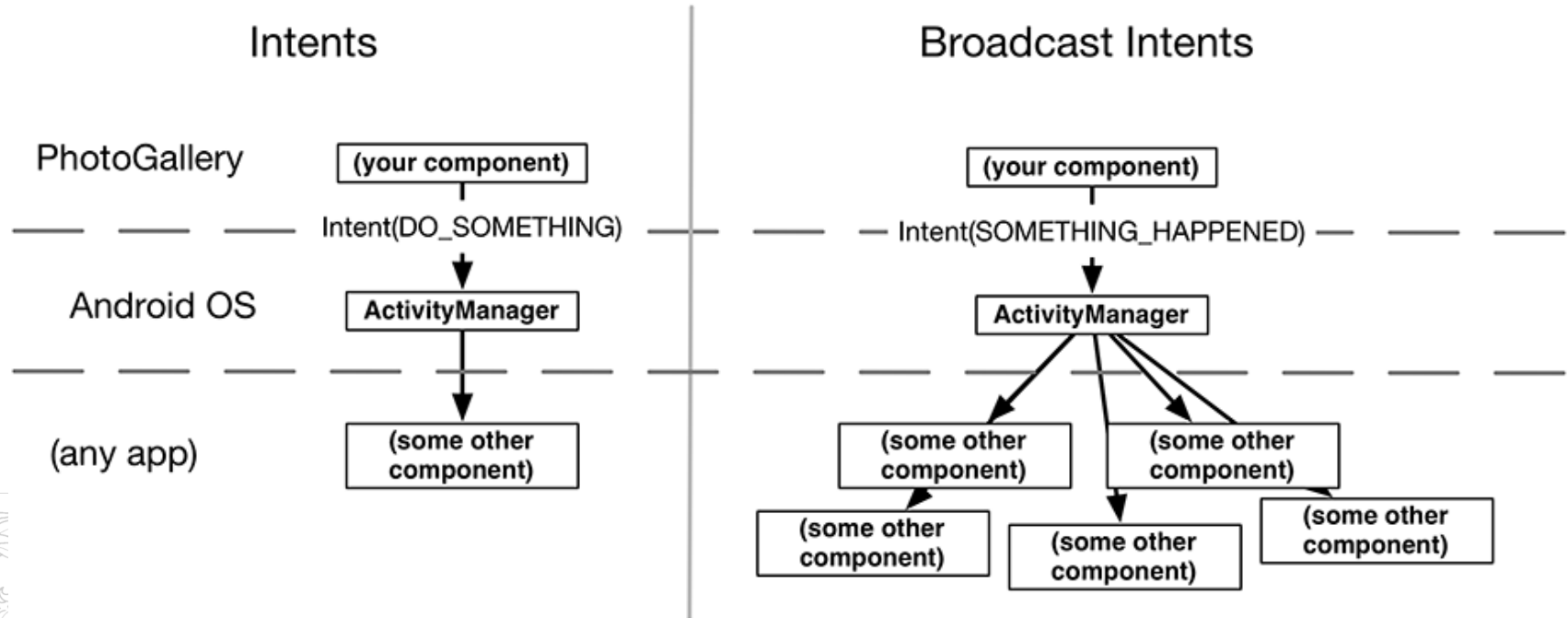
Az üzenetszórás típusai:



- Rendszer üzenetszórás
- Egyedi üzenetszórás



Intent vs Üzenetszórás

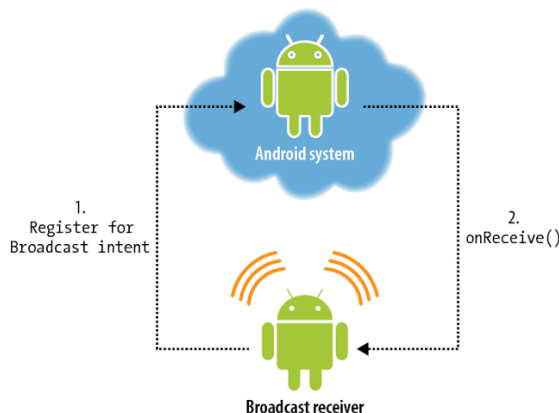


Rendszer üzenetszórás

Rendszer esemény hatására küldi ki a rendszer

Példák:

- Egy Intent, [ACTION_BOOT_COMPLETED](#) akcióval akkor lesz kiküldve amikor az eszköz elindul.
- Egy Intent, [ACTION_POWER_CONNECTED](#) akcióval akkor lesz kiküldve amikor az eszköz külső tápra lesz kötve



Egyedi üzenetszórás

Az alkalmazásunk küldi ki. Hasonló a rendszer üzenetszóráshoz.

Például az alkalmazásunk szólni szeretne más alkalmazásoknak, hogy befejezte az adatok letöltését és ezek elérhetőek.



Egyedi üzenetszórás kiküldése

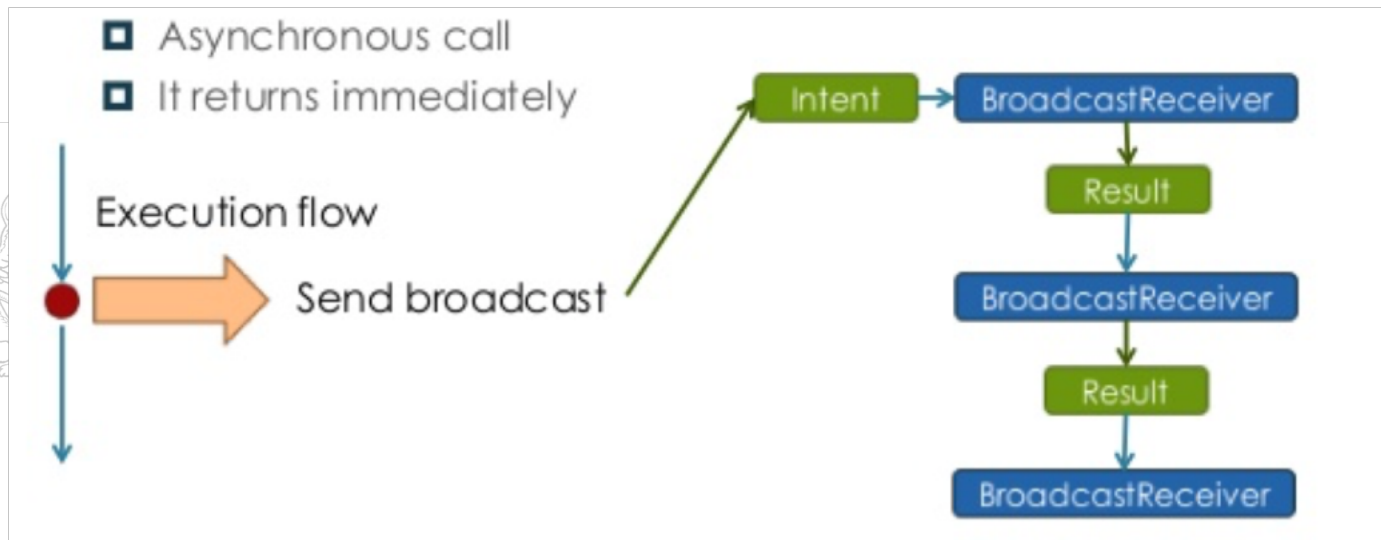
Három módon küldhetjük ki:

- Rendezett üzenetszórás.
- Normál üzenetszórás.
- Helyi üzenetszórás



Rendezett üzenszórás

- ▶ Egyszerre egy vevőnek kézbesíti `ACTION_NEW_OUTGOING_CALL`
- ▶ A `sendOrderedBroadcast()` metódussal tudjuk kiküldeni.
- ▶ A fogadók tudják továbbküldeni, vagy megállítani a láncot.
- ▶ A vevőket prioritás alapján választja ki a rendszer `android:priority` (android manifest fájlban).
- ▶ Az azonos prioritású vevők véletlenszerűen lesznek kiválasztva



Példa

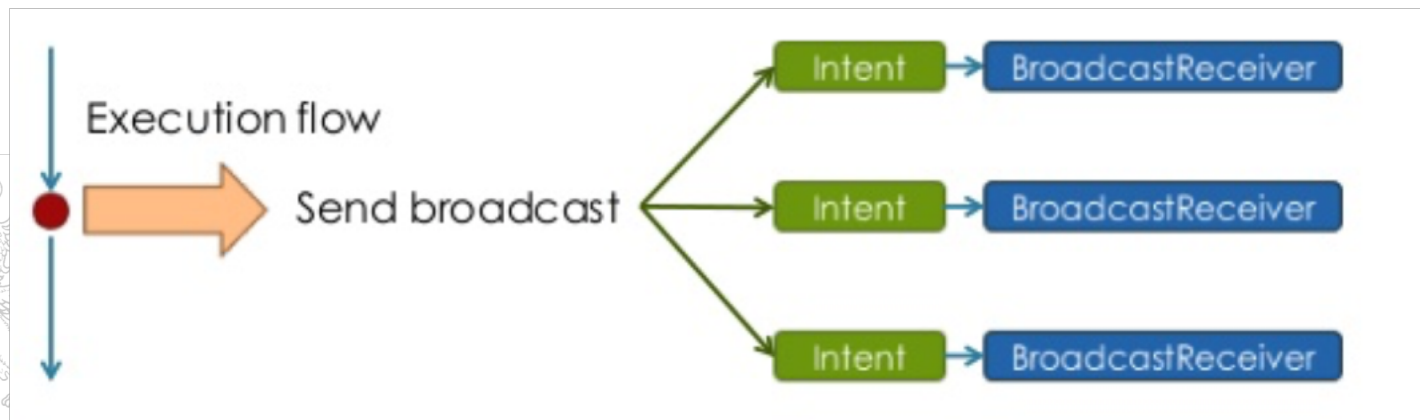
```
public class CallReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Try to read the phone number from previous receivers.
        String phoneNumber = getResultData();

        if (phoneNumber == null) {
            // We could not find any previous data. Use the original phone number in this case.
            phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
        }

        if (shouldCancel(phoneNumber)) {
            // Cancel our call.
            setResultData(null);
        } else {
            // Use rewritten number as the result data.
            setResultData(reformatNumber(phoneNumber));
        }
    }
}
```

Normál üzenetszórás

- ▶ Minden regisztrált vevőnek kézbesíti. A sorrend véletlenszerű.
- ▶ Az üzenetküldés leghatékonyabb verziója
- ▶ A vevők nem tudják továbbküldeni és leállítani.
- ▶ `sendBroadcast()` metódussal tudjuk küldeni.



Lokális üzenetszórás

- Az alkalmazáson belüli vevőknek küldi ki.
- Nincs biztonsági probléma mivel, nem processzek közötti
- Lokális üzenetszórás kiküldésére:
 - A `LocalBroadcastManager` egy példányával hívjuk meg a `sendBroadcast()`-t



```
LocalBroadcastManager.getInstance(this)
    .sendBroadcast(customBroadcastIntent);
```

Egyedi üzenetszórás

- A küldőnek és a vevőnek meg kell egyezni egy egyedi intent névben (akció név)
- Ezt definiálni kell az aktivitásban és az üzenetszórás vevőben

```
private static final String
```

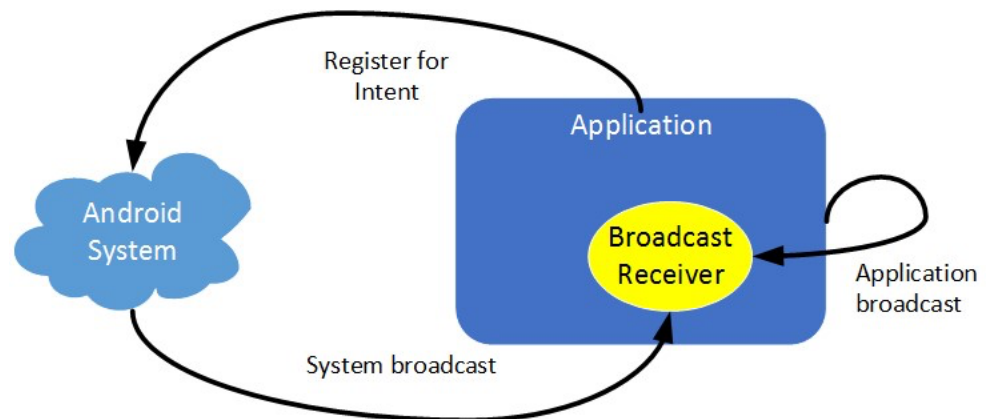
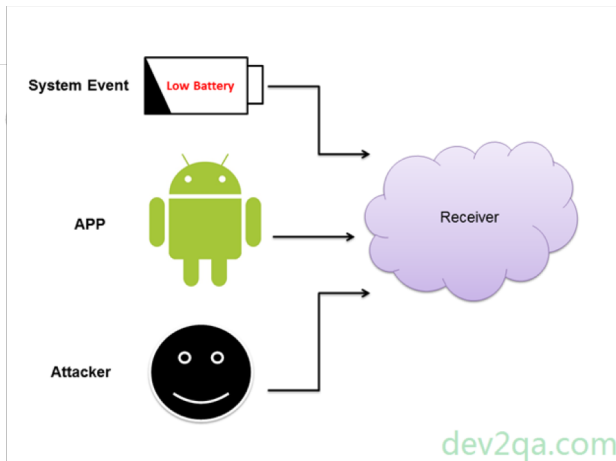
```
ACTION_CUSTOM_BROADCAST =
```

```
"com.example.android.powerreceiver.AC  
TION_CUSTOM_BROADCAST";
```



Üzenetszórás vevő

- Alkalmazás komponensek
- Regisztrálva vannak különböző üzenetszórás üzenetekre.
- Értesítést kapnak (egy `Intent`-tel):
 - A rendszertől.
 - Egy másik (vagy az adott) alkalmazástól.



■ Az üzenetszórás vevő regisztrálása

Két regisztrációs mód lehetséges:

- Statikus vevők
 - Az `AndroidManifest.xml`-ben regisztráljuk – Manifest deklarált vevő
- Dinamikus vevők
 - Az alkalmazás vagy aktivitások `context` segítségével regisztráljuk – kontextus regisztrált vevő



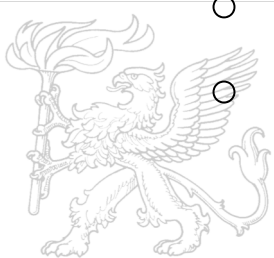
Rendszer üzenetszórás fogadása

- Az Android 8.0 (API level 26)-tól kezdve, a statikus vevők nem fogadhatják a legtöbb rendszer üzenetszórást. (le lehet írtakozni, ha nem fut akkor elindítja)
- Használjuk a dinamikus regisztrációt ezen eseményekhez



Üzenetszórás vevő

- Származtassuk le osztályunkat a BroadcastReceiver osztályból és írjuk felül az `onReceive()` metódust.
- Regisztráljuk a vevőnket és adjuk meg az intent szűrőnket:
 - Statikusan a Manifest-ben.
 - Dinamikusan, a `registerReceiver()` -rel.

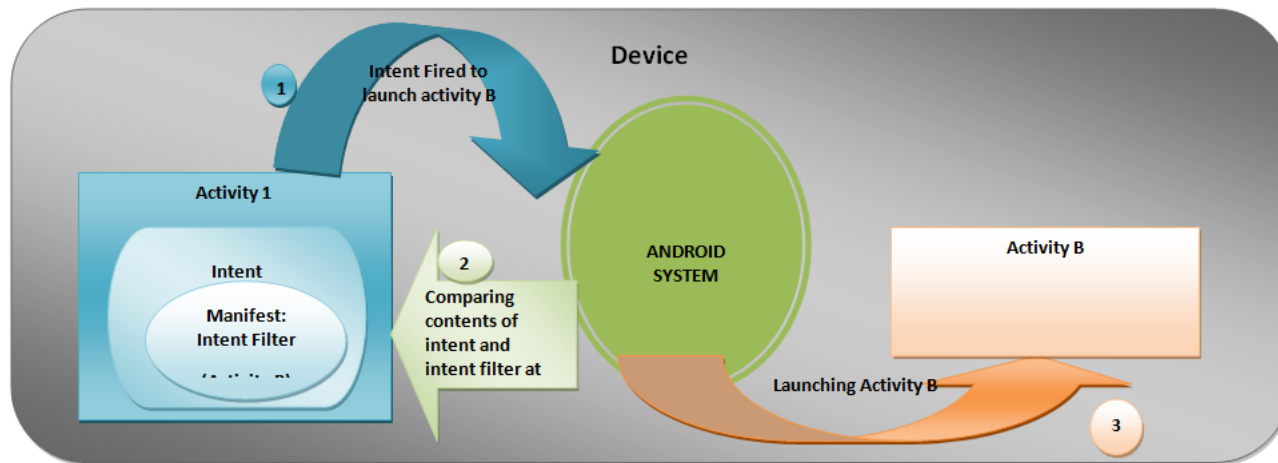


Intent szűrők

A vevő által fogadható intenteket szűri olyan tulajdonságok alapján mint pl.: `action`.

Intent szűrő megadása:

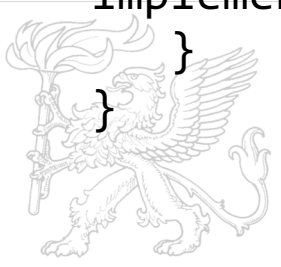
- Az `AndroidManifest.xml` fájlba, használjuk az `<intent-filter>` elemet.
- A Java fájlba `IntentFilter` objektum.



Vevő létrehozása

Android studio, **File > New > Other > BroadcastReceiver**

```
public class CustomReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // This method is called when the BroadcastReceiver  
        // is receiving an Intent broadcast.  
        throw new UnsupportedOperationException("Not yet  
implemented");  
    }  
}
```



onReceive()

```
@Override  
public void onReceive(Context context, Intent intent) {  
    String intentAction = intent.getAction();  
    switch (intentAction){  
        case Intent.ACTION_POWER_CONNECTED:  
            break;  
        case Intent.ACTION_POWER_DISCONNECTED:  
            break;  
    }  
}
```



Statikus regisztráció

```
<receiver
  android:name=".CustomReceiver"
  android:enabled="true"
  android:exported="true">
  <intent-filter>
    <action
      android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
  </receiver>
```



Dinamikus regisztráció

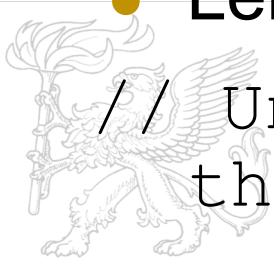
- onCreate() **vagy** onResume() -ben

```
// Register the receiver using the  
activity context.
```

```
    this.registerReceiver(mReceiver,  
filter);
```

- **Lemondás** onDestroy() **vagy** onPause().

```
// Unregister the receiver  
this.unregisterReceiver(mReceiver);
```



Lokális üzenetszórás

- ▶ Csak dinamikusan lehet regisztrálni
- A `LocalBroadcastManager` egy példánya
- hívjuk meg a `registerReceiver()`-t

```
LocalBroadcastManager.getInstance(this).registerReceiver
```

```
(mReceiver,
```

```
new IntentFilter(CustomReceiver.ACTION_CUSTOM_BROADCAST));
```



Üzenetek lemondása

```
LocalBroadcastManager.getInstance(this)  
    .unregisterReceiver(mReceiver);
```



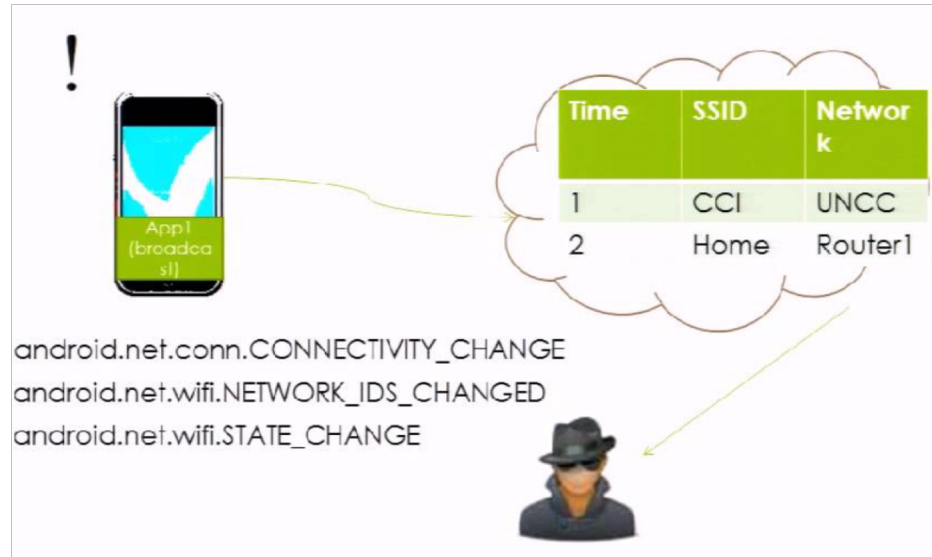
Üzenetszórás szűrése

- A szűretlen üzenetszórás biztonsági problémát okozhat.
- Pl.: Amennyiben az üzenet érzékeny információt tartalmaz azt hallgathatja egy kártékony kódot futtató helyi program



Üzenetszórás szűrése

- Amennyiben lehet akkor használjuk a LocalBroadcastManager –t amely csak a helyi kommunikációt engedi
- Használjuk a setPackage() metódust és adjuk át a csomagunk nevét. Az üzenetszórásunkat csak a csomagunkban lévő vevők kapják meg.
- Hozzáférés jogosultságok.



Küldő jogosultságok

A jogosultságok kikényszerítésére:

- Adjunk nem null értékű jogosultság argumentumot a `sendBroadcast()` metódusnak
- Csak ezen jogosultságot kérő vevők kaphatják meg az üzenetszórást



- `<uses-permission>`
`AndroidManifest.xml`

Vevő engedélyek

A jogosultságok ingyénylése:

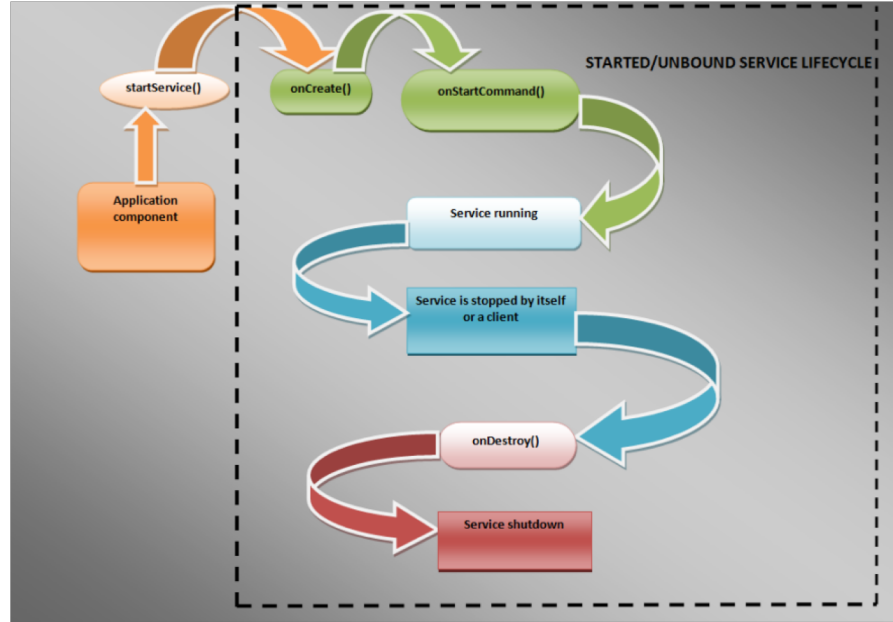
- Dinamikus esetben nem null paraméterű `registerReceiver()`.
- Egyébként `android:permission` aattribútum a `<receiver>` elemében a `AndroidManifest.xml` fájlban





Szolgáltatások

- ▶ Olyan alkalmazás komponensek amelyek hosszú futásidejű feladatokat tudnak végrehajtani a háttérben és nem rendelkeznek GUI-val.





Mire használhatóak

- Hálózati tranzakciók
- Zene lejátszása
- Fájl I/O.
- Adatbázisok kezelése.



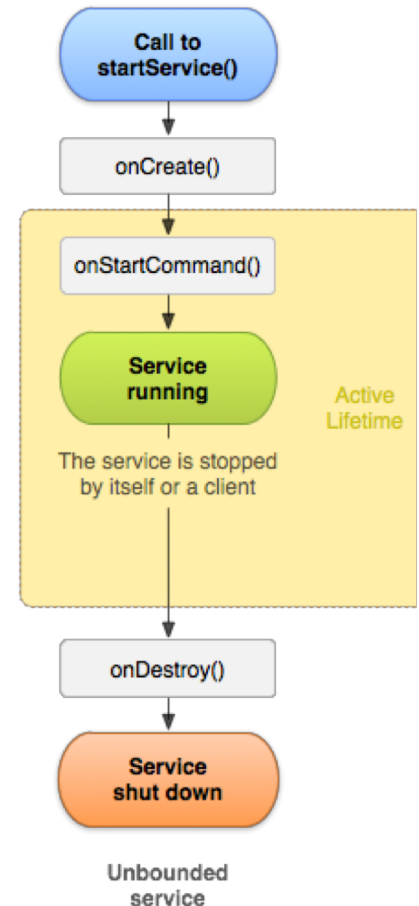
A szolgáltatás jellemzői

- Intent-tel indul
- Alkalmazás váltás közben is tovább futhat.
- Életciklus—amit menedzselnünk kell.
- Más alkalmazások is használhatják—kezeljük a jogosultságokat.
- A gazda folyamat fő szálában fut.



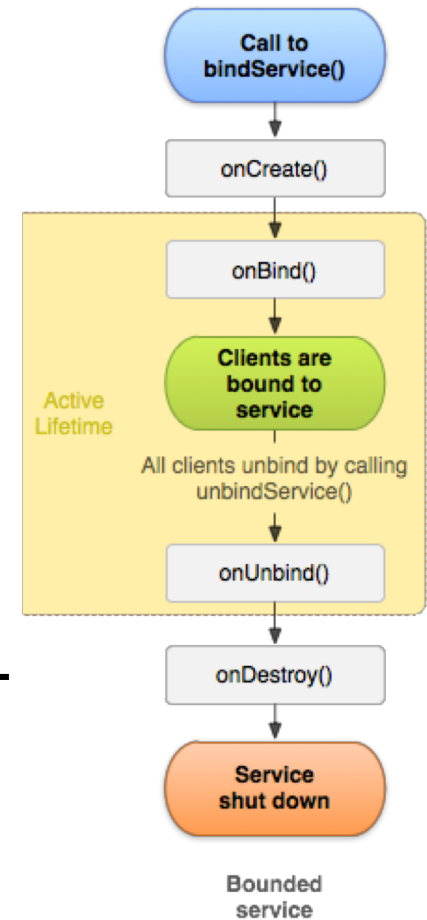
A szolgáltatás formái: started

- A `startService()`-vel indul
- Addig fut amíg le nem állítja magát
- Nem jellemző az UI frissítés



A szolgáltatás formái: kötött

- Egy kliens-szerver interfészt biztosít a komponenseknek
- A kliensek kéréseket küldenek és válaszokat kapnak
- A `bindService()`-vel indul
- Akkor áll le ha minden kliens lekapcsolódik

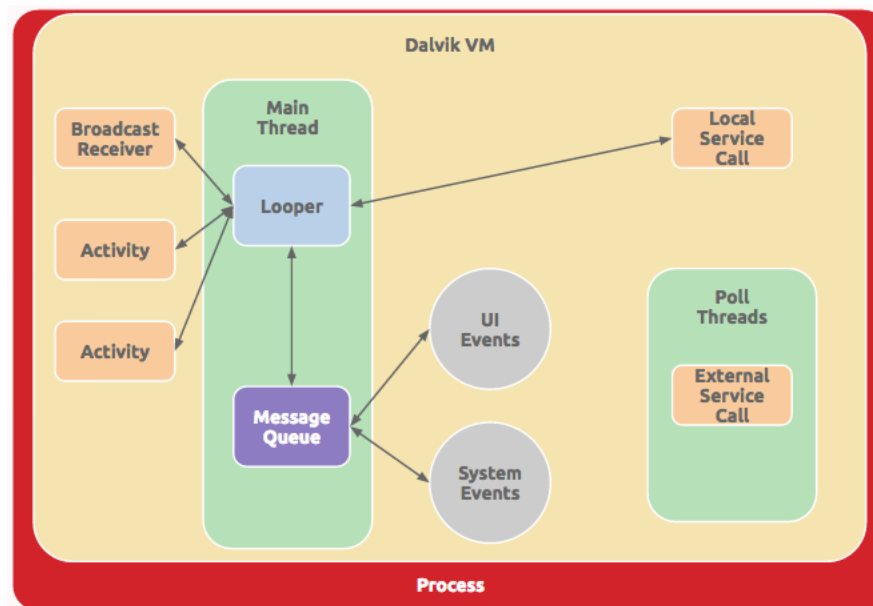




Szolgáltatások és szálak

Bár elkülönülnek az UI-tól mégis a fő szálban futnak (kivéve IntentService)

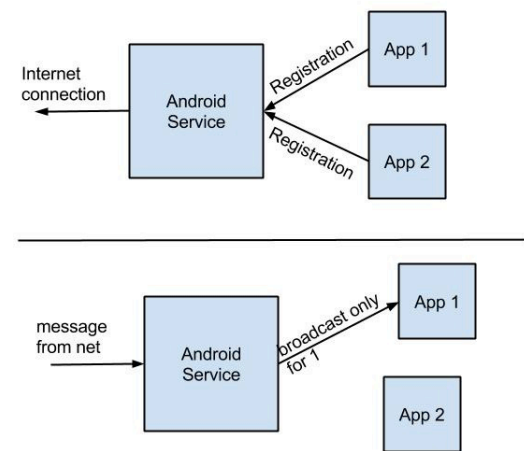
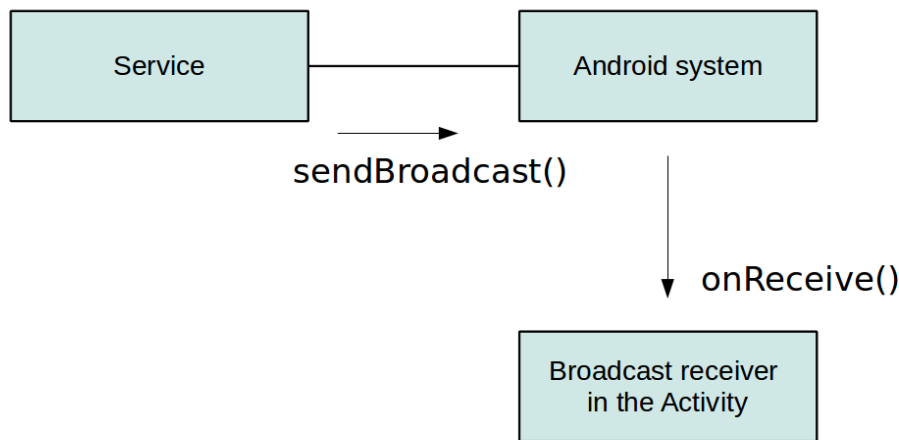
A CPU intenzív tevékenységhez indítsunk új szálát a szolgáltatásból



Az alkalmazás frissítése

Hogyan frissítsük az UI-t ha nem férünk hozzá?

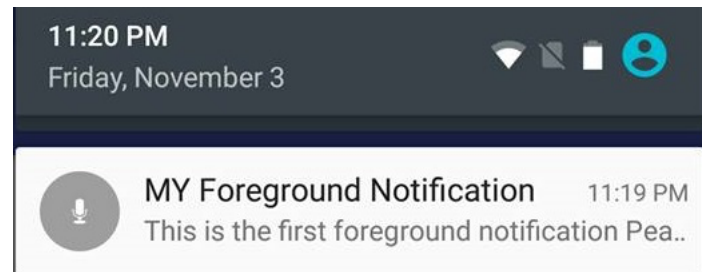
Használjunk üzenetszórást!



Előtér szolgáltatások

Bár a háttérben fut a felhasználó aktív jelenlétére van szüksége. pl.: zenelejtszó

- Magasabb prioritású mint a háttér folyamatok mivel a felhasználó érzékeli a hiányát
- A felhasználó által nem eltűnethető értesítéssel kell rendelkeznie



Háttér szolgáltatás korlátok

- Az API 26-tól, háttérben lévő alkalmazás nem hozhat létre háttér szolgáltatást.
- Az előtérben lévő app létrehozhat előtér és háttér szolgáltatást is.
- Amikor az alkalmazás háttérbe kerül akkor a rendszer leállítja a háttér szolgáltatásait
- Ezek a korlátozások nem érintik az előtér és kötött szolgáltatásokat



Szolgáltatás létrehozása

- `<service
 android:name=".ExampleService" />`
- Jogosultságok kezelése
- Alosztály az `IntentService` vagy `Service` osztályból.
- Valósítsuk meg az életciklus metódusokat
- Indítsuk el egy `Activity`.-ből
- Gondoskodjunk arról, hogy megállítható legyen



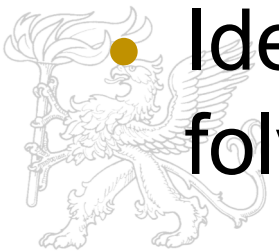
Szolgáltatás megállítása

- Egy **elindult szolgáltatásnak** kezenie kell saját életciklusát
- Ha nem áll le akkor az erőforrásokat fogja használni
- A szolgáltatásnak a stopSelf() metódussal kell magát leállítania
- Más komponens a stopService() metódussal állíthatja le
- **A kötött szolgáltatás** akkor semmisül meg ha minden kliens lekapcsolódik
- **Az IntentService** akkor semmisül meg amikor az onHandleIntent() visszatér



IntentService

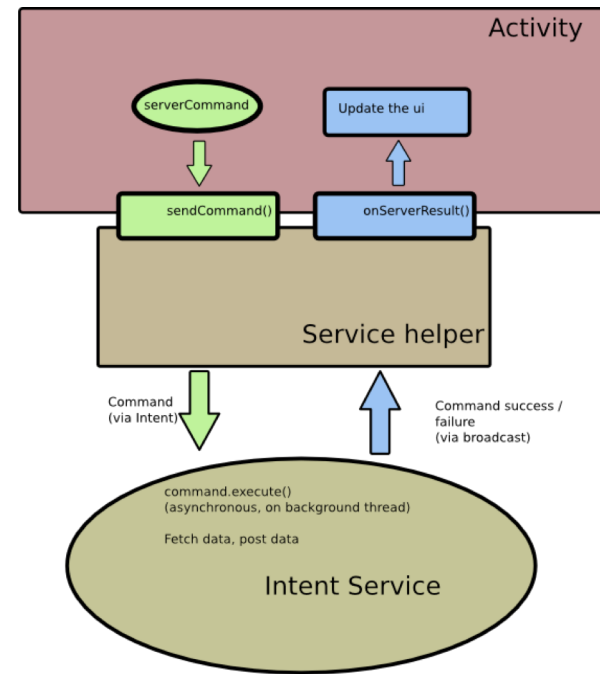
- Egyszerű szolgáltatás, egyszerűsített életciklussal
 - Dolgozó szálakat használ
 - Leállítja magát ha befejezte a munkát
-
- Ideális az egy szálon futó hosszú háttér folyamatokhoz





IntentService korlátok

- Nem fér hozzá az UI-hoz
- Csak egy kérést tud egyszerre teljesíteni
- Nem szakítható meg

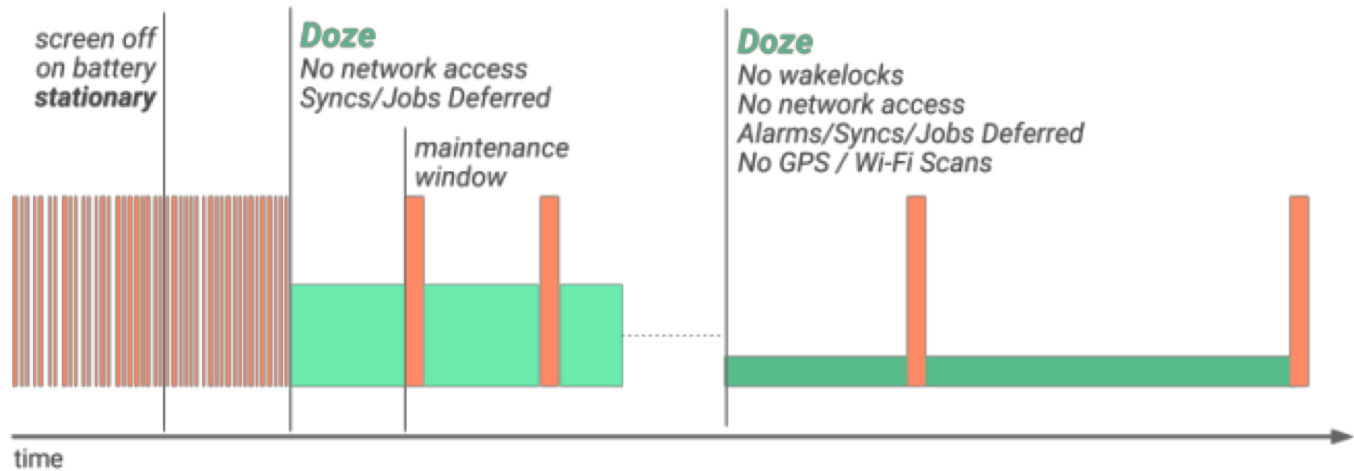
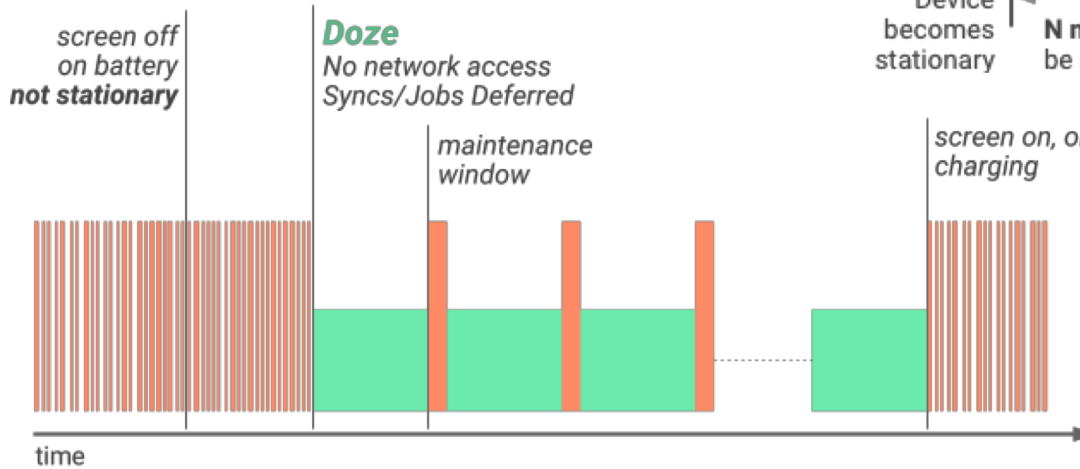
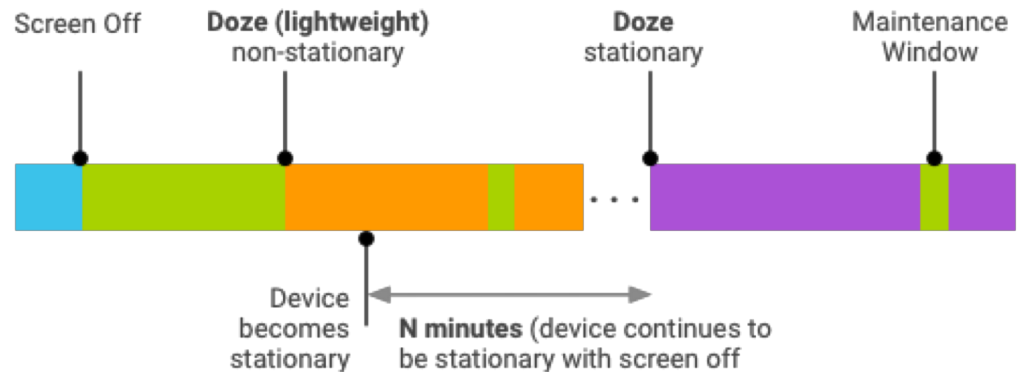


IntentService megvalósítás

```
public class HelloIntentService extends IntentService {  
    public HelloIntentService() { super("HelloIntentService");}  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        try {  
            // Do some work  
        } catch (InterruptedException e) {  
            Thread.currentThread().interrupt();  
        }  
    } // When this method returns, IntentService stops the service, as  
    appropriate.  
}
```



Doze



A mai előadás tartalma

- ▶ Üzenetszórás
 - Áttekintés
 - Egyedi üzenetszórás
 - Üzenetszórás vevők
 - Üzenetszórás vevők megvalósítása
 - Üzenetszórás szűrése
 - Legjobb gyakorlat
- ▶ Szolgáltatások



A következő előadás tartalma

- ▶ Értesítések
- ▶ Riasztások

