

Android

Komponens közti kommunikáció, Intent,
BroadcastReceiver

Dr. Ekler Péter
peter.ekler@aut.bme.hu



Department of
Automation and
Applied Informatics

Quiz time



<https://kahoot.it/#/>

Komponens közí kommunikáció

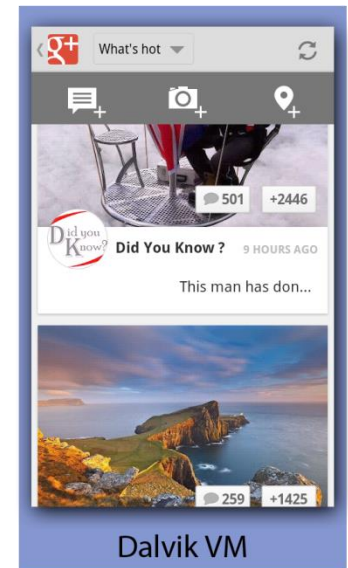
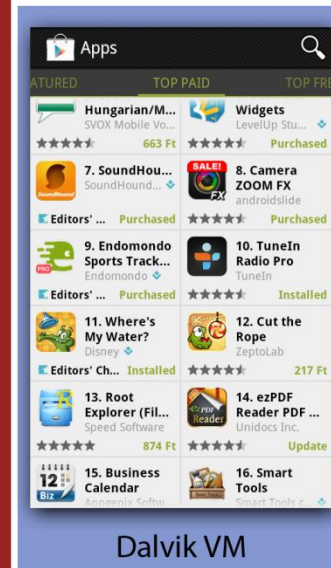
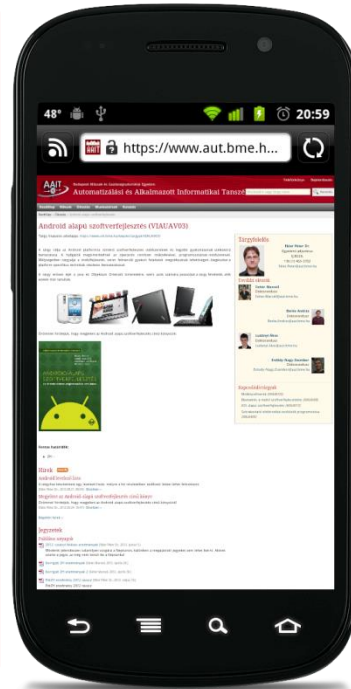
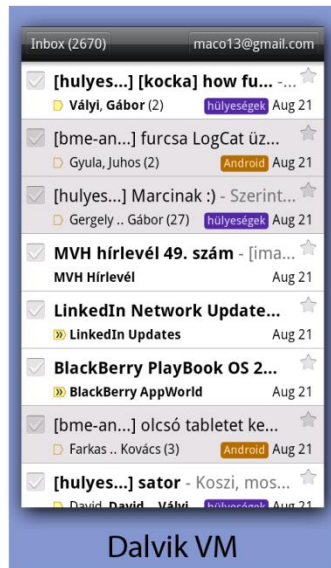
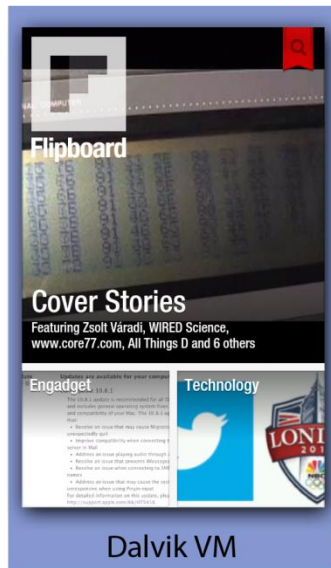
Bevezetés

- A legtöbb platformon az alkalmazások egymástól elkülönítve futnak
 - > Minden app a saját „homokozójában” (*Sandbox*)
 - > Szigorú korlátozások a sandbox-ból kinyúló műveletekre
 - Hardver elérés, pl kamera, szenzorok, stb
 - Rendszerszintű adatok, háttértár
 - Szálak, alk. komponensek közti kommunikáció
 - > Cél: adatvédelem, alkalmazások védelme egymástól

Bevezetés

Mi a helyzet Androidon?

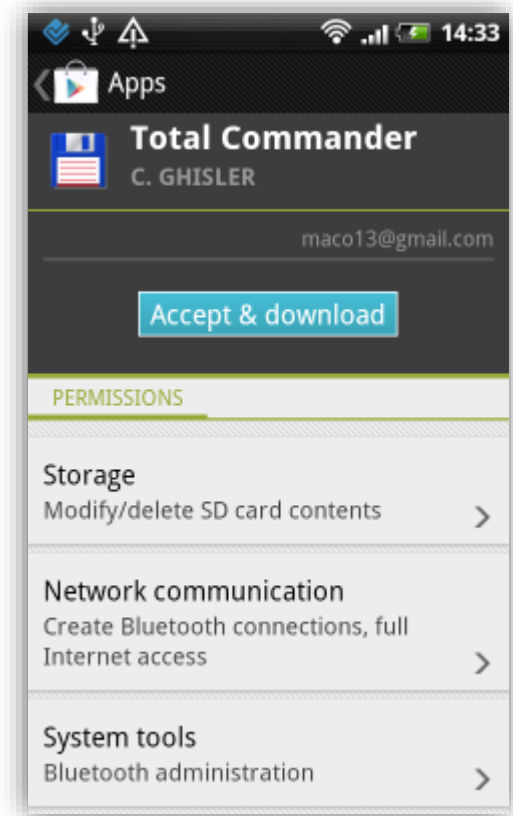
- Alkalmazások külön Dalvik VM példányokban (ez is sandbox)



Bevezetés

Mi a helyzet Androidon?

- Alkalmazások külön Dalvik VM példányokban (ez is sandbox)
- Kritikus műveletekhez engedély szükséges



Bevezetés

Mi a helyzet Androidon?

- Alkalmazások külön Dalvik VM példányokban (ez is sandbox)
- Kritikus műveletekhez engedély szükséges
- Alkalmazás = komponensek halmaza

A komponensek akár alkalmazások között is kommunikálhatnak egymással (!)

- Két komponens között: *Intent*
- Egy komponensből mindenki másnak: *Broadcast Intent*
- Csak adat megosztása (ContentProvider)

Kommunikáció formái 1/3

- Egyik komponensből a másikba: *Intent*

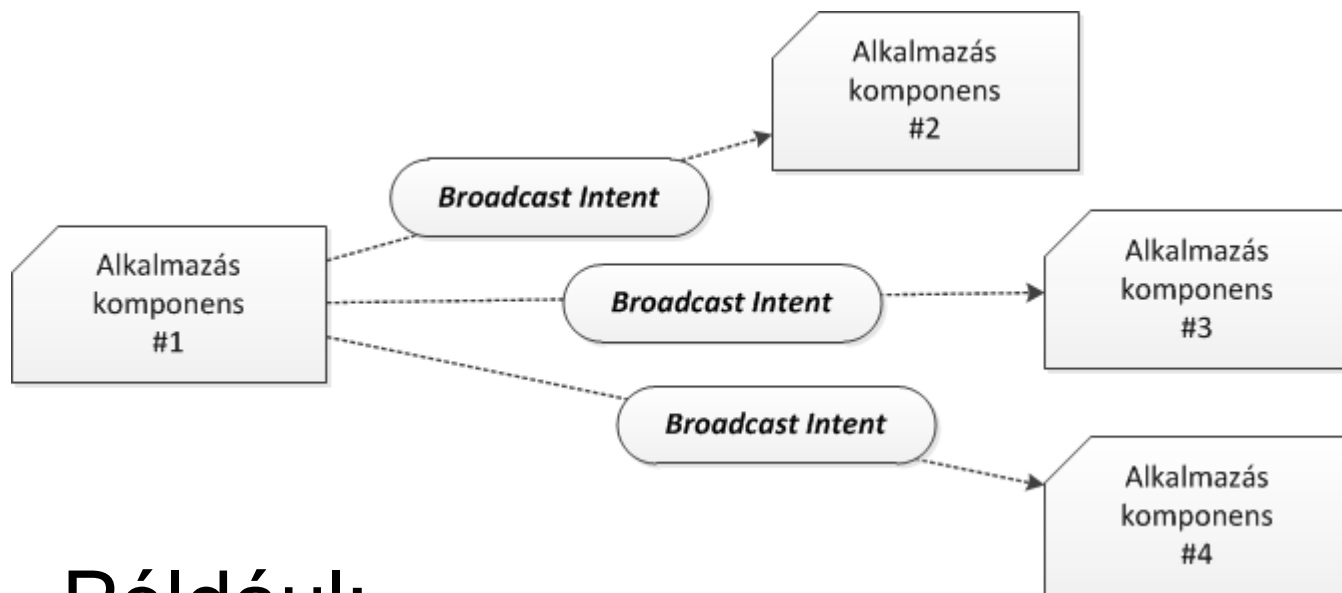


Például:

- Következő képernyőre lépés (új Activity indítása)
- Zenelejátszó service indítása

Kommunikáció formái 2/3

- Egy komponensből mindenki másnak: **Broadcast Intent**



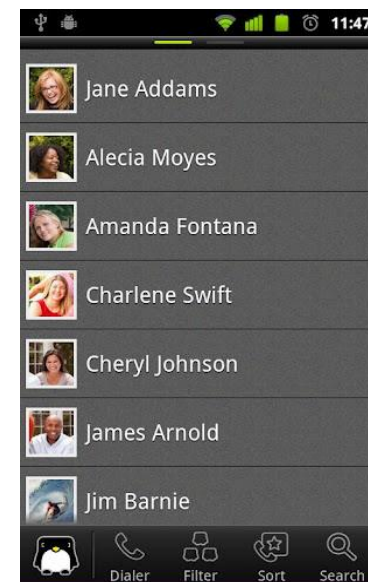
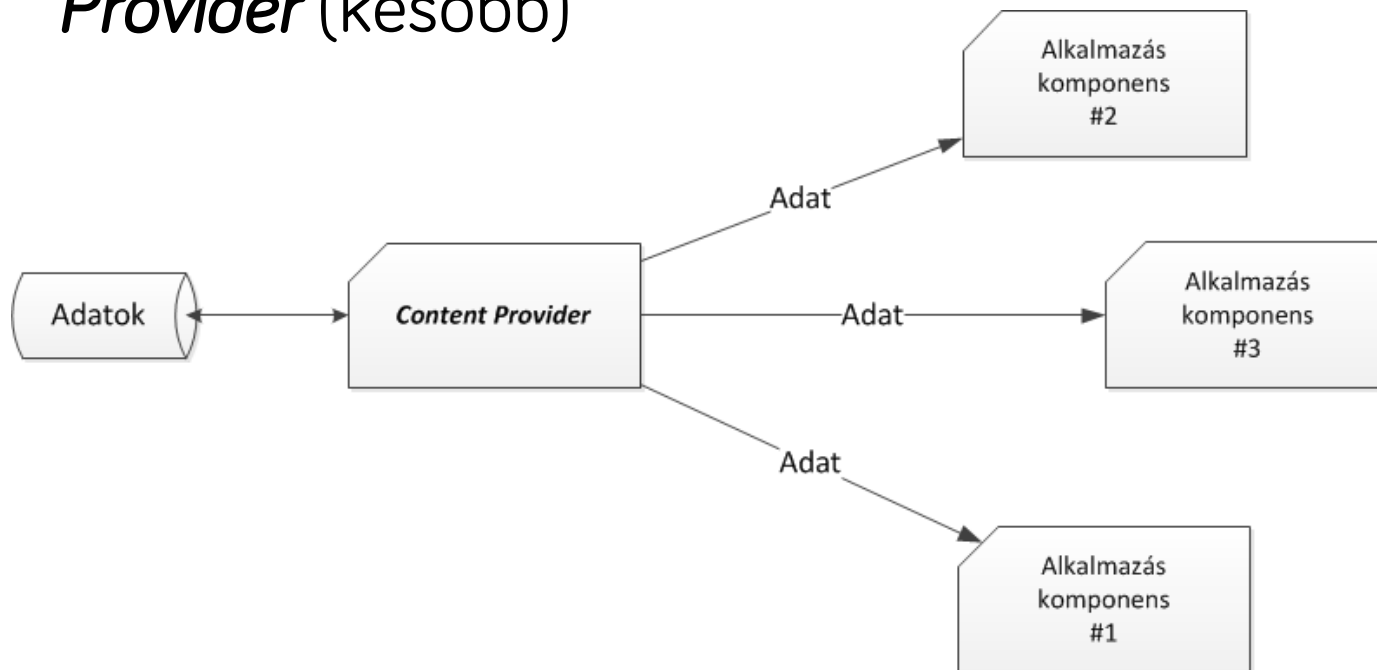
Például:

- „Akkufeszültség alacsony” rendszerüzenet

- `val intentStart: Intent = Intent(this, „hu.bme.aut.demoapp.DetailActivity”)`
- `startActivity(intentStart)`

Kommunikáció formái 3/3

Adatok szolgáltatása komponensek közt: *Content Provider* (később)

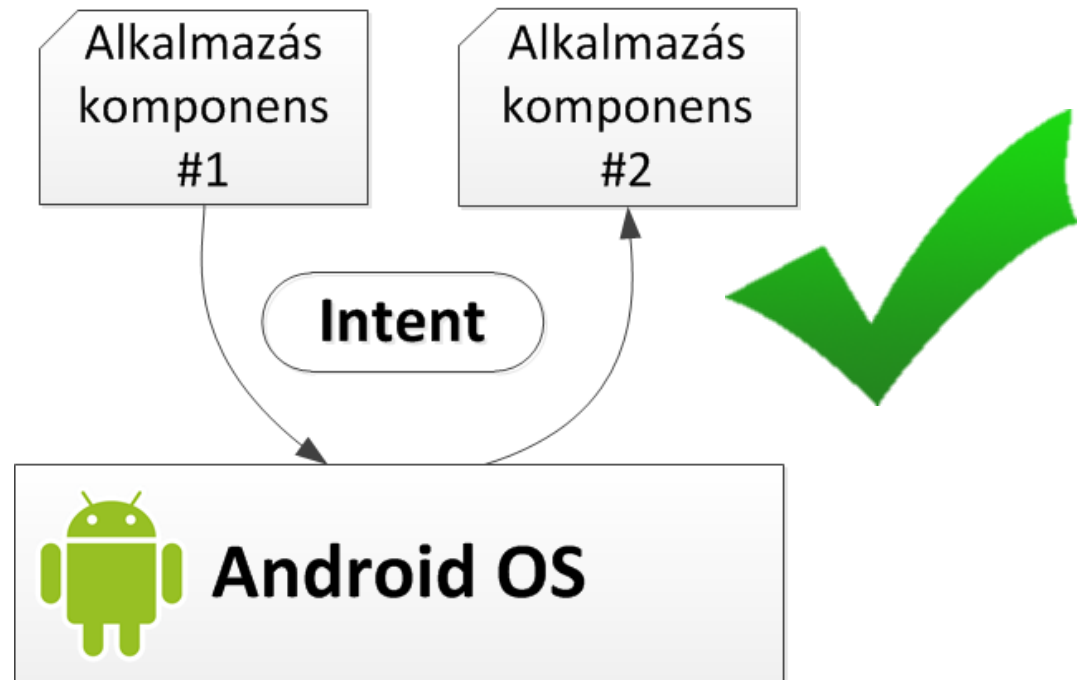
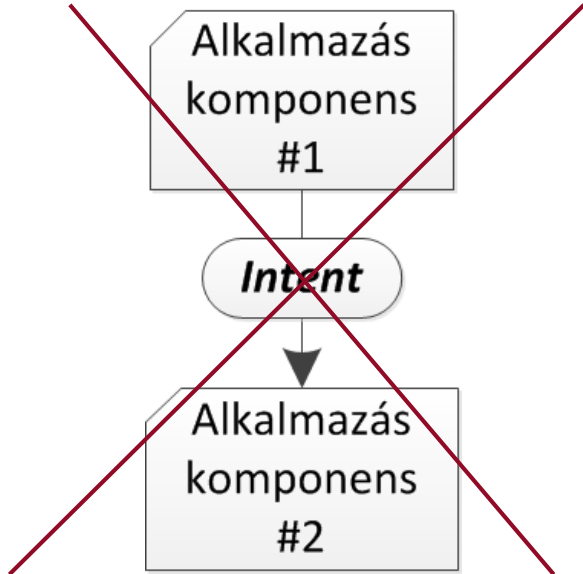


Például:

- Névjegyzék elérése saját alkalmazásból

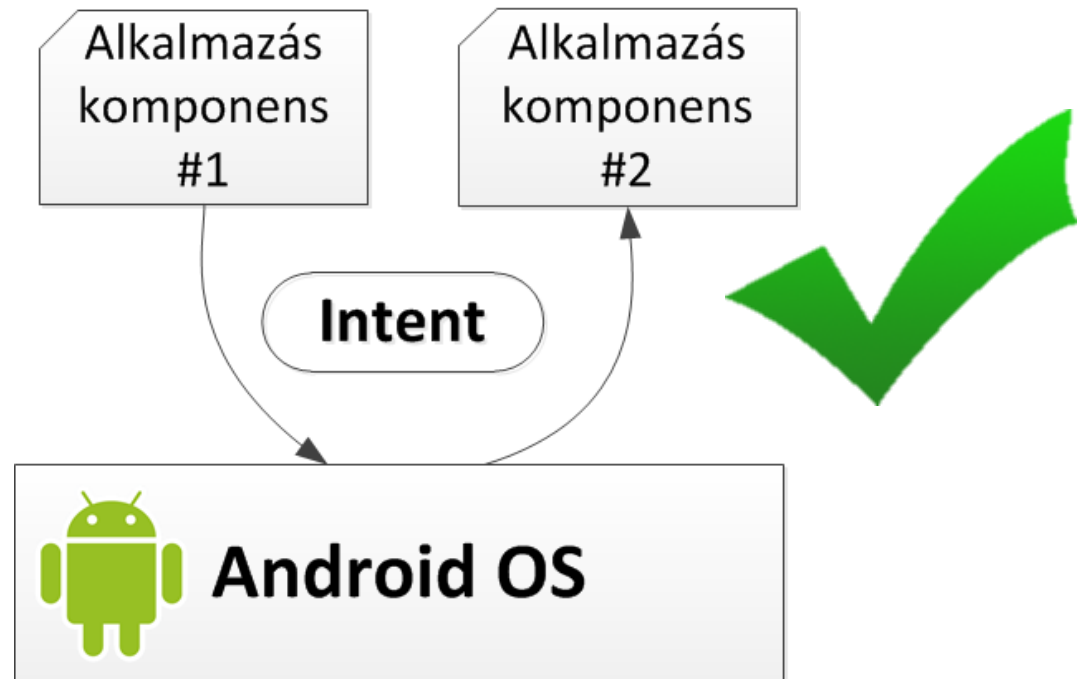
Intent átadása

- Mindig az Android runtime-on keresztül!



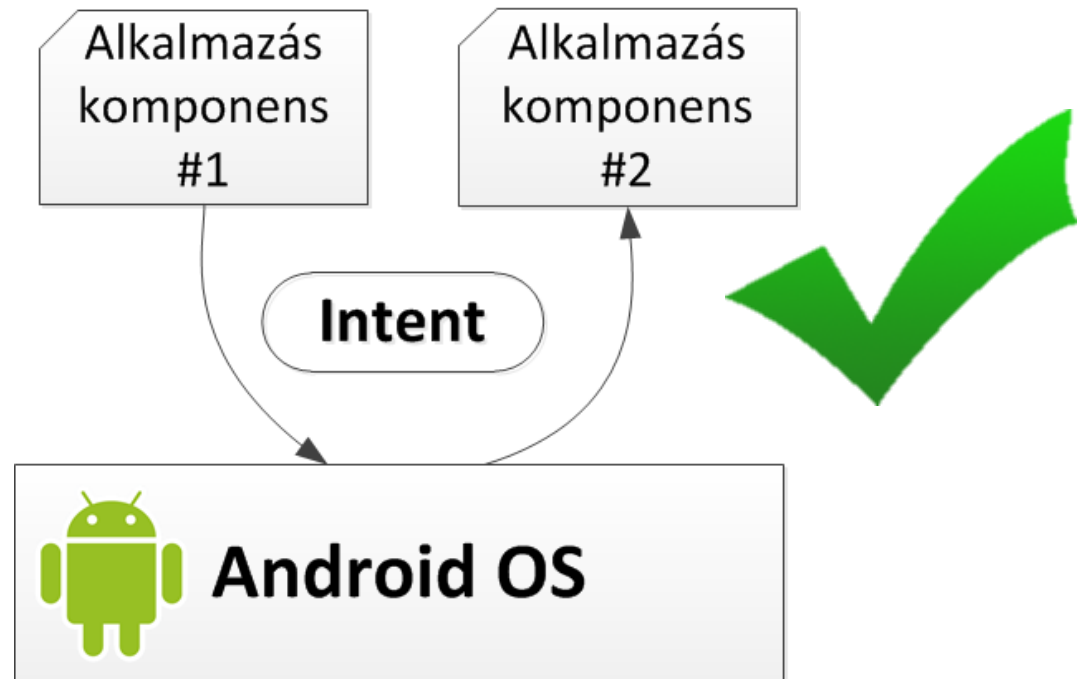
Intent átadása

- Mindig az Android runtime-on keresztül!



Intent átadása

- Mindig az Android runtime-on keresztül!



intent

Intent (*szándék*)

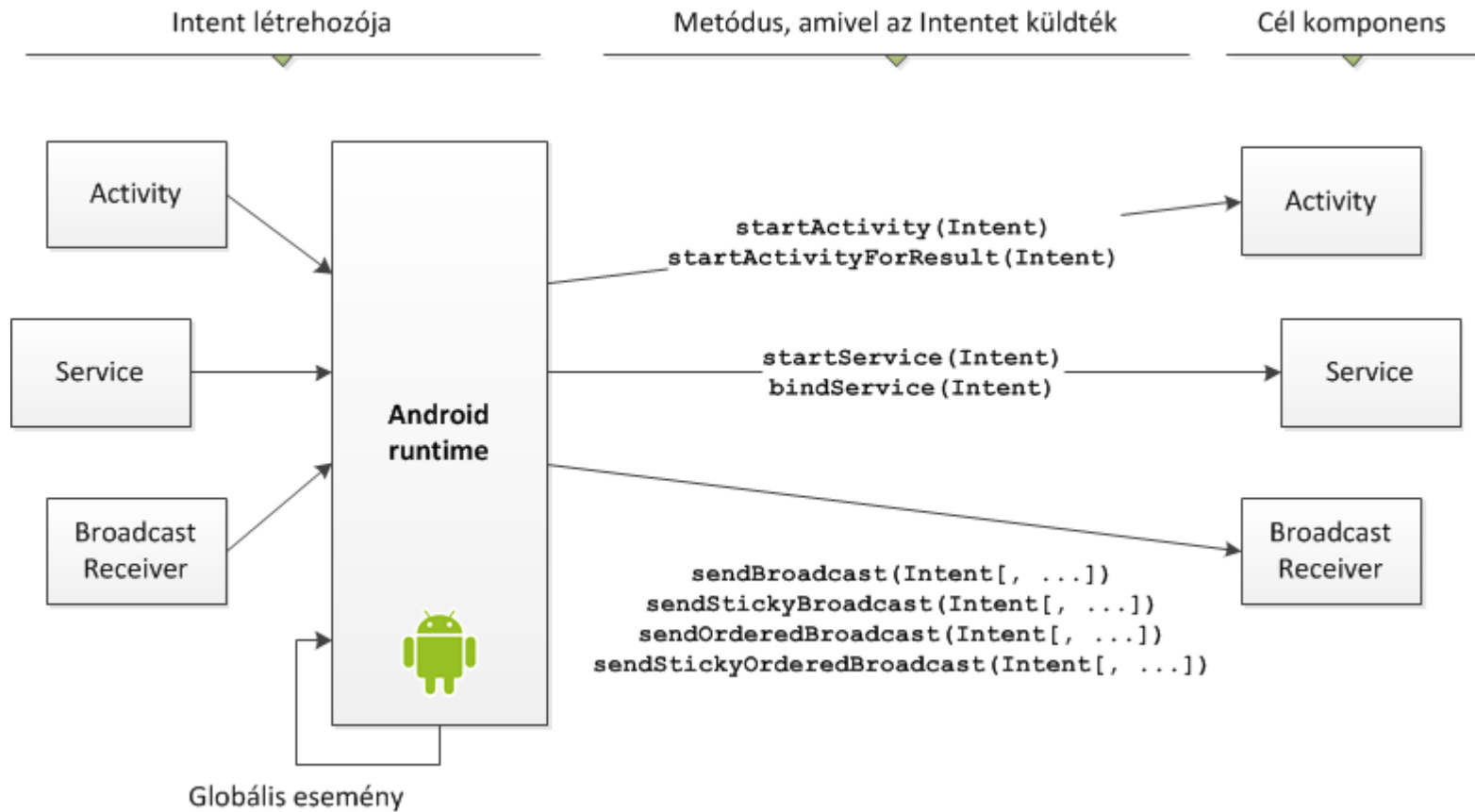
- Passzív adatstruktúra (~struct)
- Késői (futás idejű) kötést valósít meg alkalmazás komponensek között
 - > Komponensek: Activity, Service, Broadcast Receiver
- Az elvárt vagy bekövetkezett esemény absztrakt leírása
 - > Elvárt esemény leírása, ha az Intent hatására történik valami
 - Activity, Service, Broadcast Receiver regisztrálása / aktiválása
 - > Bekövetkezett esemény leírása, ha az Intent valamilyen esemény hatására jön létre
 - Broadcast üzenet (főleg rendszer események)

Intent kézbesítés

- Az Intent objektum tehát háromféle komponensnél köthet ki:
 - > **startActivity[ForResult](Intent)** : Activity indítása vagy folytatása
 - > **startService(Intent)** ,
bindService(Intent) : Service indítása vagy futásidejű kötés megvalósítása
 - > **send[Ordered|Sticky]Broadcast(Intent)** : minden érdekelt **BroadcastReceiver(BR)**-hez eljut

Intent kézbesítés

- Az Android mindhárom esetben megkeresi a megfelelő címzettet, és példányosítja ha szükséges
- Egy Intent objektum mindig csak egyféle komponenshez jut el
 - > *startActivity()*-vel átadott Intent-et nem kapják meg a Service-ek és a BR-ek
 - > Rendszerszintű esemény értesítését (Broadcast Intent) csak BR-ek kapják
 - > *startService(Intent)* csak Service-hez kerülhet



Intent részei

- **Címzett komponens osztályneve** (*Component name*): ha üres akkor az Android megkeresi a megfelelőt
- **Akció** (*Action*): az elvárt vagy megtörtént esemény
- **Adat** (*Data*): az adat (URI-ja és MIME típusa), amin az esemény értelmezett
- **Kategória** (*Category*): további kritériumok a feldolgozó komponessel kapcsolatban
- **Extrák** (*Extras*): saját kulcs-érték párok, amiket át akarunk adni a címzettnek
- **Kapcsolók** (*Flags*): Activity indításának lehetőségei

Felépítése



Intent felépítése

- Intent = Adatcsomag
- Információkat tartalmaz...
 - > ...az operációs rendszernek, ami eldönti hogy milyen komponenst és hogyan kell indítani/folytatni
 - Komponens neve
 - Category
 - Flags
 - > ...és annak a komponensnek ahova kézbesítődik
 - Action
 - Data
 - Extras

Intent használata

- Leggyakoribb használata: új Activity indítása
- Két módon lehetséges
 - > **Explicit** – ismerjük a hívandó Activity osztálynevét (tipikusan alkalmazáson belül)
 - > **Implicit** – nem tudjuk / nem akarjuk eldönteni, hogy melyik Activity szükséges, csak azt hogy mire akarjuk használni (tipikusan másik alkalmazást használunk)
 - Pl. kép megjelenítése, PDF megnyitása, videó lejátszása, névjegy kiválasztása, stb...

Explicit Intent

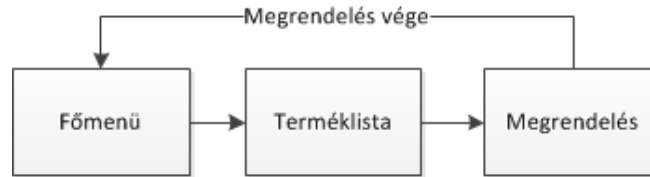
- Mindkét esetben a *startActivity()* függvényt használjuk:
 - > **startActivity(Intent)**
- **Explicit hívás:** az Intent-ben kitöltjük a címzett komponens nevét (konstruktorból vagy setterrel)

```
var i: Intent = Intent(getApplicationContext(),  
                           ListProductsActivity.class)  
startActivity(i)
```

- Ha a **ListProductsActivity**-ből már van példány a memóriában akkor folytatódik, ha nincs akkor az Android példányosítja és elindítja

Activity visszatérése

- Egy Android alkalmazás általában több Activity-ből épül fel, amik egymást indítják



- Gyakran szükséges visszajelzés arról, hogy a hívott Activity hogyan fejeződött be
 - > Melyik névjegyet választotta a felhasználó?
 - > Történt megrendelés?
 - > Bejelentkezés sikeres?

startActivityResult

- **startActivity()** -vel indítva nem kapunk visszajelzést
- Megoldás:
`startActivityResult(Intent, requestCode)`
- Több ilyen is lehet egy Activity-ben, a **requestCode** nevű integer különbözteti meg őket
- Az így indított Activity-k befejeződésük után vissza tudnak jelezni a hívónak
 - > **finish()** előtt **setResult(requestCode)** a hívott oldalon

onActivityResult

- Visszatérési érték kezelése a hívó oldalon (callback):

```
onActivityResult(requestCode, resultCode, extras) {...}
```

- Paramétereit:
 - > **requestCode**: integer, ugyanaz mint a **startActivityForResult**-ban megadott

onActivityResult

```
onActivityResult(requestCode, resultCode,  
extras) {...}
```

> **resultCode**: integer, eredmény számkódja

- **Activity.RESULT_OK** (= -1)
- **Activity.RESULT_CANCELED** (= 0, ezzel tér vissza akkor is, ha a Vissza gombra nyomott a user)
- Sajátot is definiálhatunk, például:

```
public static int RESULT_ORDER_SUCCESS  
= 2;
```

```
public static int RESULT_LOGIN_OK = 3;
```

```
public static int RESULT_LOGIN_FAIL =  
4;
```

Miért static?

onActivityResult

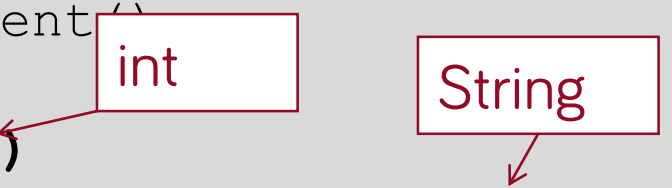
onActivityResult(requestCode, resultCode, extras) {...}

- > **extras**: ha nem elég a resultCode, akkor egy Intent objektumot is visszaadhatunk, amiben adatot helyezünk el
 - Intent Extras: tetszőleges kulcs-érték párok
 - Egy Intent objektumban akármennyi elhelyezhető
 - *Kulcs*: String, aminek prefixe a package name
 - *Érték*: akármilyen beépített típus, de lehet saját is, ha megvalósítja a Serializable vagy a Parcelable interfészt

Intent Extras

- Feltöltése: **intent.putExtra (name, value) ;**

```
var resultIntent: Intent = Intent()  
resultIntent.putExtra(  
    "UserID", currentUser.getId())  
resultIntent.putExtra("role", currentUser.getRole())  
setResult(RESULT_OK, resultIntent)  
finish()
```

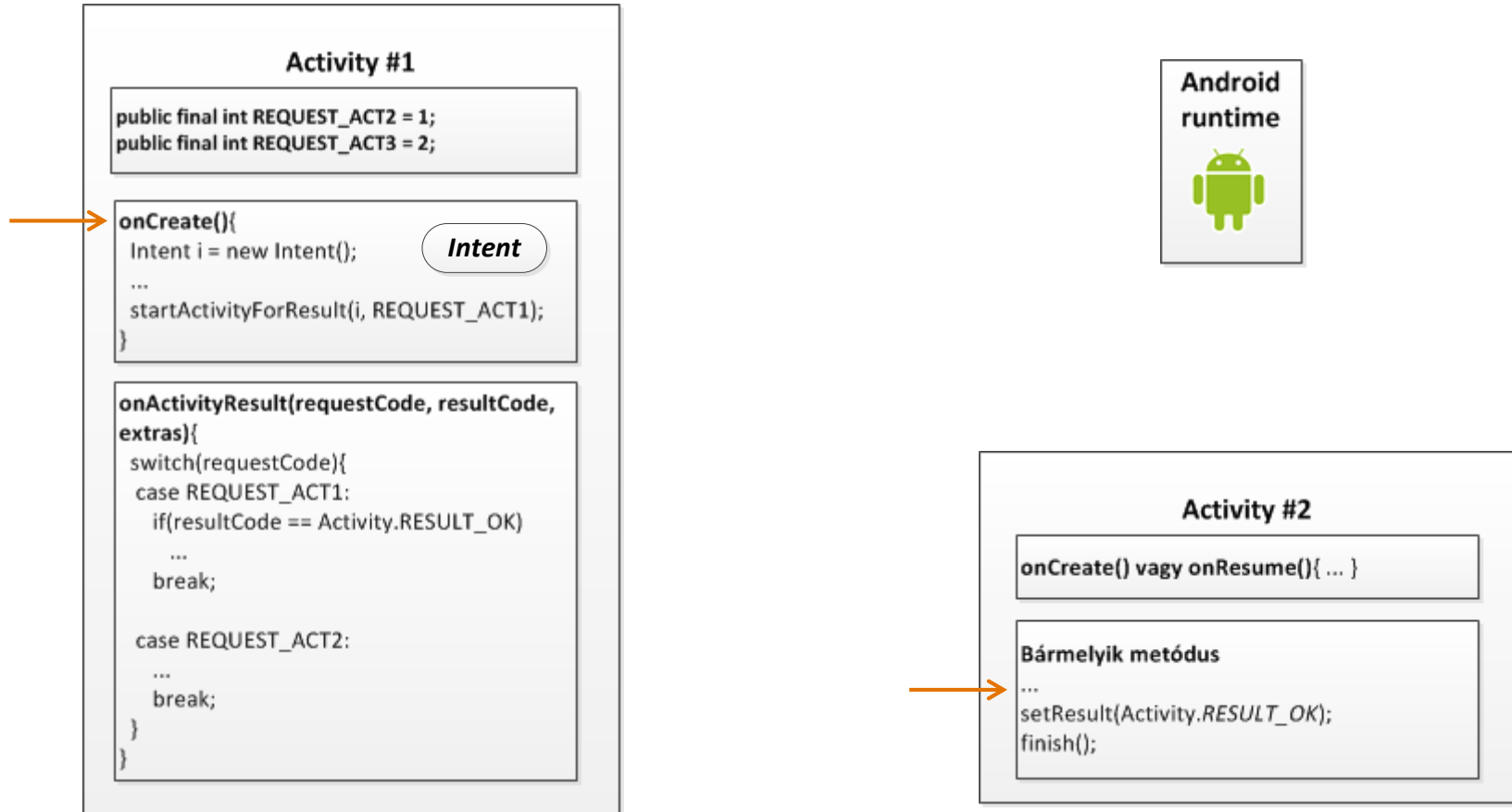


Intent Extras

- Lekérdezése:

```
intent.get[Típusnév]Extra(name[, defaultValue])
```

Activity visszatérése



Implicit Intent

- Implicit hívás: azt mondjuk meg, hogy milyen akció történjen
 - > Ha szükséges, akkor azt is, hogy milyen adat(ok)on
- Hívás gomb megnyomásának szimulálása:

```
var i: Intent = Intent(Intent.ACTION_CALL_BUTTON)
startActivity(i)
```

Implicit Intent - Példa

- Telefonszám felhívása

```
var i: Intent = Intent(Intent.ACTION_DIAL,  
                        Uri.parse („tel:0630-123-4567”))  
startActivity(i)
```

Akció

Adat (URI)

- Névjegy kiválasztása

```
var i: Intent = Intent(Intent.ACTION_PICK,  
                        ContactsContract.Contacts.CONTENT_URI)  
startActivity(i)
```

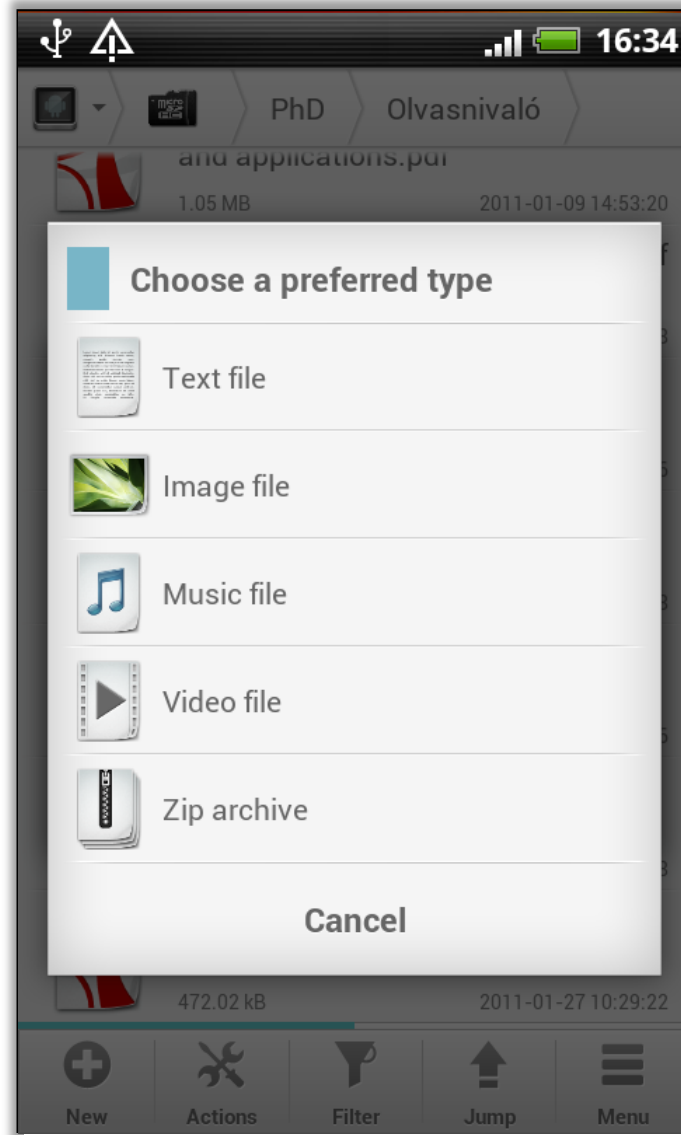
Akció

Adat (URI)

Implicit Intent - Példa

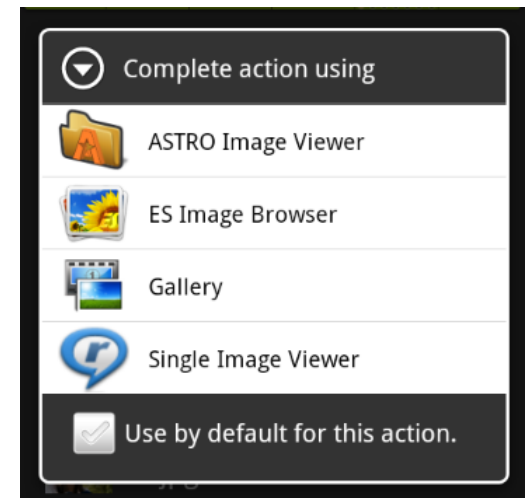
- Pdf megnyitása

```
var sdCard: File = Environment.getExternalStorageDirectory()
var pdf:File = File(sdCard+"/docs", "sample.pdf")
var i: Intent = Intent(Intent.ACTION_VIEW)
i.setDataAndType(Uri.fromFile(pdf), "application/pdf")
startActivity(i)
```



Implicit Intent – Több célpont

- *ActivityNotFoundException*, ha nem talál megfelelőt
- Amennyiben több alkalmazás is képes a kért akcióra: „*Complete action using*” dialógusablak
 - Ha nincs default megadva, akkor a felhasználó választ
 - Egyébként indul az alapértelmezett alkalmazás
 - (új app telepítése után újra rákérdez az alapértelmezésre)



Implicit Intent – Akciók

- Nem csak megnyitni és kiválasztani lehet
- Rengeteg „beépített” akció, a fontosabbak:
 - > *ACTION_EDIT*: szerkesztés, pl. névjegy, txt, kép
 - > *ACTION_DELETE*: adat törlése
 - > *ACTION_WEB_SEARCH*: Google keresést indít a kapott szövegre
 - > *ACTION_CALL*: kapott telefonszám felhívása
 - > *ACTION_PICK*: kiválasztás listából, pl. névjegy, fénykép, zene, videó
 - > *ACTION_VIEW*: megnyitás a megfelelő alkalmazással, pl: telepítő fájl, pdf, txt, fénykép

Implicit Intent – Beépített akciók

A majdnem teljes lista (új verziónál általában bővül!):

ACTION_MAIN
ACTION_VIEW
ACTION_ATTACH_DATA
ACTION_EDIT
ACTION_PICK
ACTION_CHOOSER
ACTION_GET_CONTENT
ACTION_DIAL
ACTION_CALL

ACTION_SEND
ACTION_SENDTO
ACTION_ANSWER
ACTION_INSERT
ACTION_DELETE
ACTION_RUN
ACTION_SYNC
ACTION_PICK_ACTIVITY
ACTION_SEARCH

ACTION_WEB_SEARCH
ACTION_FACTORY_TEST

Implicit Intent

Az Intent nem csak arra jó, hogy a következő képernyőre ugorjunk

Intent általánosan: *„Kérés egy akció elvégzésére valamilyen adaton”*

- startActivity esetén nem fogalmazunk meg sem akciót, sem adatot, csak a cél komponenst.
- *„Az elvárt vagy bekövetkezett esemény absztrakt leírása”*

Implicit Intent

- Ha az Intentben nem kötelező megadni a célpontot (Activity-t), akkor honnan tudja az Android hogy mit indítson?
- Az Intent objektum információkat tartalmaz arról, hogy mit kell csinálni
 - > *Action, Data, Category* mezők
- Az Android megkeresi a legjobb célpontot
- Neve: Intent feloldás (*Intent Resolution*)

Intent feloldás

- Az Intent-ben lévő Action, Data és Category mezők tartalma alapján
- Mindháromra teszteli az összes alkalmazás összes komponensét
- Regisztráció Intentek kezelésére: Intent filterek segítségével
 - > Egy XML node az AndroidManifest-ben
 - > Ezzel történik a szolgáltatások kiajánlása is más komponensek / alkalmazások számára

Intent Filter

- Lehetséges a saját alkalmazásunk funkcióinak kiajánlása mások számára
 - > Az Androidban beépítve vannak ilyenek, ld. Intent Action (pl. ACTION_CALL, ACTION_IMAGE_CAPTURE)
- Az AndroidManifest-ben kell deklarálni (miért?)
- Ha nincs Intent filter beállítva, akkor a komponens kizárólag explicit intentet képes fogadni
- Ha van Intent filter, akkor explicit és implicit intenteket is ki tud szolgálni

Intent Filter

- Egy komponenshez több filtert is beállíthatunk
 - > Mindegyik leírja a komponens egy képességét, amivel a hozzá érkező implicit intenteket kezelni tudja
 - > Explicit intentek mindenképp eljutnak a komponenshez, függetlenül a beállított filterektől
 - > Minden filter egy kijánlott funkciónak, képességnek felel meg
 - > Pl: névjegy szerkesztő activityhez két intent filter:
 - Kiválasztott névjegy szerkesztése
 - Új (üres) névjegy létrehozása
 - > A megfelelő Activity/Service/BR node-ban **<intent-filter>** node bevezetése

Intent Filter – Action

- Action: az az akció, amit a komponens le tud kezelni
 - > Vagy egy default ACTION
 - Pl. „android.intent.action.ACTION_CAPTURE_IMAGE”, ha egyedi kamera alkalmazást csinálunk
 - > Vagy teljesen egyedi név kell
 - Package name prefix

```
<action android:name="com.amorg.notepad.SHOW_NOTE" />
```

- Legalább egy Action kell az intent filterbe
- Ha az Intent-ben nincs kitöltve az Action, akkor bármilyen action-el rendelkező komponens átmegy a teszten

Intent Filter – Data

- Milyen típusú **adat** kezelésére képes a komponens
- Data mező részei:
 - > **mimeType**: megszokott MIME típus (pl. text/*), de egyedit is definiálhatunk. Ekkor a package name előtt vnd (*Vendor*) prefix szükséges
`<data android:mimeType="vnd.amorg.notepad.note/*"/>`
 - > **URI**: scheme://host:port/path
- Intent URI és type nélkül csak ugyanilyen filterre illeszthető
- Intent csak URI-val: csak type nélküli, min. egy illeszthető URI-val rendelkező filterre (pl. *mailto:* vagy *tel:*)
- Intent csak type-al: A filter is csak egy megfelelő type-ot ír elő

Intent Filter – Category

- Category: milyen körülmények között szolgálható ki az Action. Fontosabbak:
 - > **LAUNCHER**: csak Activity-re állítható, ekkor megjelenik az alkalmazások között a launcher-ben (így lehet több belépési pont, lásd Google+ kliens)
 - > **HOME**: csak Activity-re állítható, ha be van állítva és nincs Action, akkor a natív home screen alternatívája lesz (így lehet saját home képernyőt csinálni)
 - > **DEFAULT**: Data és Action beállítással együtt érvényes, alapértelmezett akciót állít be az adat típuson (pl. kép, videó, pdf, doc)

Intent feloldás

- Csak implicit intent esetén (!)
- Sok lépésből álló folyamat
- Az Intentnek mindhárom (*Action*, *Data*, *Category*) teszten meg kell felelnie!
- Hogy juthat el mégis a komponenshez?
 - > Explicit intent
 - > Ugyanennek a komponensnek van egy másik intent filtere, amin átment

Gyakoroljunk!

- Készítsünk egy alkalmazást, amely alapértelmezett alkalmazás *txt* állományok kezelésére!

Intent FLAG-ek

- Új Activity alapértelmezetten a Back Stack tetejére jön létre
- Ha szükséges, változtathatunk ezen Intent Flag-ek segítségével
- Ezek módosítják az alapértelmezett viselkedést (pl. a Back gomb hatását), használatuk csak indokolt esetben javasolt, és tesztelni kell a használhatóságot! (pl. böngészőkben mit csinál a Back?)

Intent FLAG-ek

- A leggyakrabban használtak:
 - > **FLAG_ACTIVITY_NEW_TASK**: A létrehozott Activity nem a hívó taszkban indul, hanem újat kap (launcher jellegű appok)
 - > **FLAG_ACTIVITY_SINGLE_TOP**: Ha a hívott Activity-ből már van egy példány a stack tetején, akkor nem hoz létre egy újabbat, hanem a meglévőt folytatja
 - > **FLAG_ACTIVITY_CLEAR_TOP**: Ha a hívott Activity már ott van valahol a stack-ben, akkor letakarítja a fölötte lévőket és azt folytatja

Intent FLAG-ek

- Beállítása:

```
var i:Intent = Intent(Intent.ACTION_VIEW)
i.setFlags( Intent.FLAG_ACTIVITY_CLEAR_TOP |
            Intent.FLAG_ACTIVITY_NEW_TASK )
```

- Nagyon ritka az az eset, amikor Intent Flag-et kell használni
- Rengeteg más Flag van még, lásd a setFlags() metódus dokumentációjánál

Intent lekérése

- Az Activity/Service/BR lekérheti azt az Intentet, ami őt indította
 - > Extrák kibányászása
 - > Data kinyerése
 - > (minden más részére is van getter)
- **getIntent()** metódus szolgáltatja

```
var dataUri: Uri = intent.getData()  
var extras: Bundle = intent.getExtras()  
var action: String = intent.getAction()
```

Linkify

- Automatikusan linket készít a TextView-ban lévő szövegrészekből
- Rákattintva a megfelelő Intentet küldi a rendszernek

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/linkify_me"
    android:autoLink="phone | email"
/>
```

Linkify saját típusok

- Beépített típusok
 - > Web, email, phone, map
- Saját típus is definiálható, amiből link készül
 - > RegExp + séma megadásáva



Ebből gyártja majd az Intent data URI-t

- > Mi lesz az Intent Action?
 - android.intent.action.VIEW

Pending Intent

- Intentet nem csak azonnal lehet küldeni
- Előre definiálhatunk olyan Intentet, ami majd később, vagy akár egy másik alkalmazásból fog küldődni
 - > Pl. widgetre kattintáskor vagy időzítve küldődik
- Azért, hogy előre felkészülhessünk a fogadására
- Neve: PendingIntent

Intent - összefoglaló

- Android alkalmazás komponensek:
 - > Activity, Service, Broadcast Receiver
- Kommunikáció köztük: Intentekkel
- Nem csak alkalmazáson belül, hanem azok között is lehetséges
 - > Használhatunk más alkalmazásban lévő komponenst
 - > Kijánlhatjuk a sajátunkat

BroadcastReceiver komponens

Broadcast események

- Rendszerszintű eseményekre fel lehet iratkozni – Broadcast üzenet
- Az Intent alkalmas arra hogy leírja az eseményt
- Sok beépített Broadcast Intent, lehet egyedi is

ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED

ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN

Broadcast események

- Nem csak az Android, hanem alkalmazások (Activity-k és Service-ek) is dobhatnak Broadcast Intentet
 - > Telephony service küldi az ACTION_PHONE_STATE_CHANGED Broadcast Intentet, ha a mobilhálózat csatlakozás megváltozott
 - > android.provider.Telephony.SMS_RECEIVED
 - > Sok más, érdemes tájékozódni ha valamit szeretnénk lekezelni
 - > Saját alkalmazásunkból is dobhatunk a **sendBroadcast(String action)** metódussal
 - > Ez is Intent, lehet Extra és Data része

Broadcast intentek elkapása

- Broadcast Receiver nevű komponens segítségével
 - > Kódból vagy manifestben kell regisztrálni
 - (bizonyos Action-ök esetén nem mindegy, tájékozódni!, pl. `TIME_TICK`)
 - > Intent filterrel állíthatjuk be hogy milyen Intent esetén aktivizálódjon
- Nem Activity, nincs felhasználói felülete
- Azonban képes Activity-t indítani
- Használata: `BroadcastReceiver` osztályból származtatunk, és felüldefiniáljuk az `onReceive()` metódust, majd intent-filter

Broadcast intentek kezelése

```
class OutgoingCallReceiver : BroadcastReceiver() {  
    override fun onReceive(context: Context, intent: Intent) {  
        val outNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER)  
        Toast.makeText(context, outNumber, Toast.LENGTH_LONG).show()  
    }  
}
```

AndroidManifest.xml:

```
<receiver android:name=".OutgoingCallReceiver">  
    <intent-filter>  
        <action android:name=  
            "android.intent.action.NEW_OUTGOING_CALL"/>  
    </intent-filter>  
</receiver>
```

Broadcast intentek kezelése

Broadcast továbbdobásának megakadályozása:

- `abortBroadcast()`

Például ha a fülhallgató média gombjait kell kezelni és nem akarjuk, hogy a zenelejátszó is megkapja a Broadcast-ot 😊

Gyakoroljunk!

- Készítsünk egy AirPlane mód változásra figyelő BroadcastReceivert!
- Készítsünk egy kimenő hívásra figyelő BroadcastReceivert!
 - > Változtassuk meg a hívott számot!
 - > Egészítsük ki SMS figyeléssel!
 - Valósítsuk meg, hogy ne kerüljön be inbox-ba a bejövő SMS

SMS Receiver 1/2

- Manifest:

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

```
...
```

```
<receiver android:name=".SMSReceiver" android:enabled="true">
```

```
    <intent-filter android:priority="1000">
```

```
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
```

```
    </intent-filter>
```

```
</receiver>
```

SMS Receiver 2/2

```
class SMSReceiver : BroadcastReceiver() {  
    override fun onReceive(context: Context, intent: Intent) {  
        val extras = intent.extras ?: return  
        val pdus = extras.get("pdus") as Array<ByteArray>  
        for (pdu in pdus) {  
            val msg = SmsMessage.createFromPdu(pdu)  
            val origin = msg.originatingAddress  
            val body = msg.messageBody  
            Toast.makeText(context,  
                "SMS caught, number: $origin body: $body", Toast.LENGTH_LONG)  
                .show()  
        }  
    }  
}
```

Alkalmazáskomponens indítása Boot után

- Néha olyan szolgáltatásokra van szükség, amelyek mindig futnak a készüléken
- Ilyen esetben fontos, hogy a készülék indítása esetén ezek automatikusan is el tudjanak indulni
- Az Android lehetőséget biztosít arra, hogy feliratkozzunk a „*Boot befejeződött*” eseményre és valamilyen alkalmazás komponens elindítsunk:
 - > *BroadcastReceiver* definiálása Manifest-ben
 - > *android.intent.action.BOOT_COMPLETED*
- A *BroadcastReceiver onReceive()* függvényében elindíthatjuk a megfelelő komponens

Pending Intent

- Intentet nem csak azonnal lehet küldeni
- Előre definiálhatunk olyan Intentet, ami majd később, vagy akár egy másik alkalmazásból fog küldődni
 - > Pl. widgetre kattintáskor vagy időzítve küldődik
- Azért, hogy előre felkészülhessünk a fogadására
- Neve: PendingIntent

Hogy is volt?

- Hogy kell Activity-t indítani, ha vissza akarunk kapni adatot belőle?
 - > `startActivityForResult()`
- Mit kell beállítani a manifestben, ha saját Home Screen-t akarunk csinálni?
 - > Intent filter: `category=HOME`
- Mit csinálhatunk a rendszerszintű események bekövetkezésekor?
 - > Bármit

Kérdések

