Android

További Fragment és RecyclerView lehetőségek, Komponens közi kommunikáció, Intent, BroadcastReceiver

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

BME AUT
Department of Automation and Applied Informatics

Tartalom

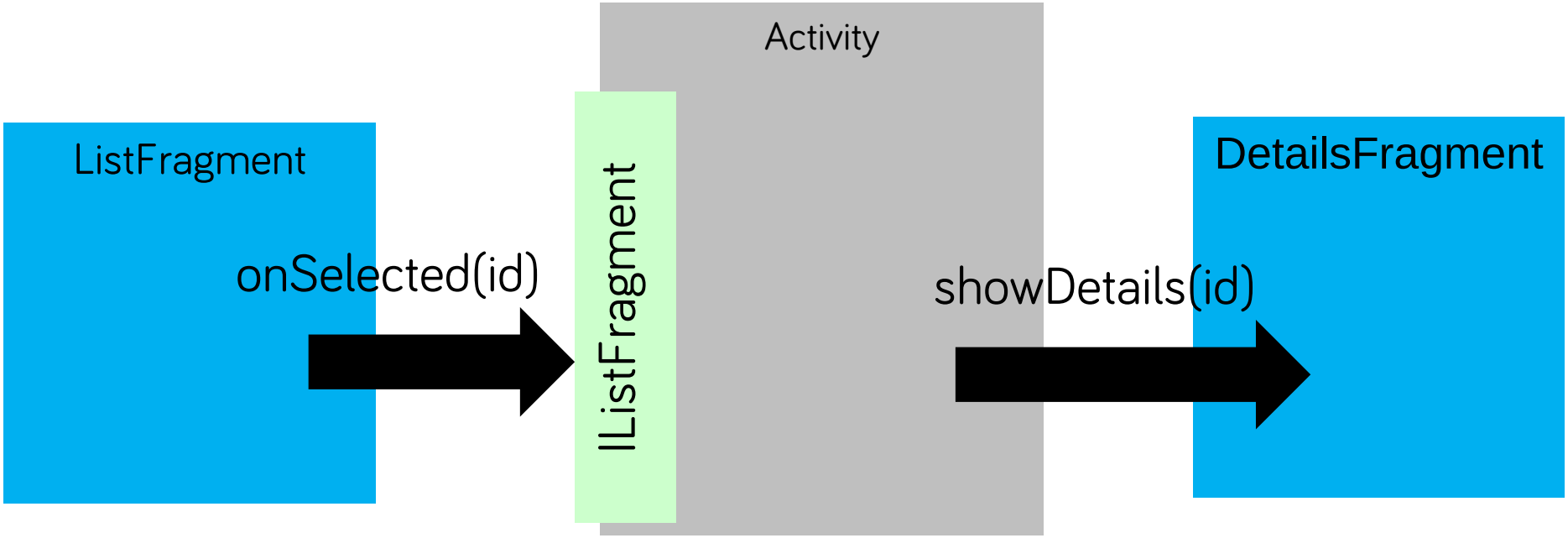
- További Fragment lehetőségek
- További RecyclerView lehetőségek
- Komponens közi kommunikáció
- Intent

További Fragment lehetőségek

DialogFragment, ViewPager

Fragment kommunikáció I.

- Egy Fragment-nek egységbezártnak kell lennie
  => közvetett kommunikáció
  - > Az Activity közvetít



Fragment kommunikáció II.

- Ha mégis szükség van a közvetlen Fragment-kommunikációra:
  - > Fragment.targetFragment propertyvel állítható/elérhető
- Az így létrejövő kapcsolat túléli a forgatásokat is

Fragment paraméterek

- Szükség lehet paraméter átadás
  - pl. Melyik cikk részleteit jelenítse meg a hírolvasó, stb..
- Első ötlet
  - ~~Mi inicializáljuk – Konstruktor paraméter~~
  - Nem csak mi inicializálhatjuk! – pl. Elforgatás
- Használjuk a Factory pattern-t
  - Fragment.arguments:Bundle property
  - Mentésre kerül a Fragment példánnyal

Fragment paraméterek

- Egy Fragmentet a publikus, paraméter nélküli konstruktorával példányosítunk
- Ha paramétereket kell átadnunk:
  - Példányosításkor Bundle-ben adjuk át őket
    - .setArguments(...) (Kotlinban arguments property)
  - Inicializáláskor ezt a Bundle-t kérjük el
    - .getArguments()
  - A forgatást is túléli az Arguments Bundle

DialogFragment I.

- Egy Fragment dialógusként is megjelenhet
  - Dialógus egyedi layout-tal
  - Az AlertDialog.Builder továbbra is használható
- Így egy dialógus is ugyanolyan életciklussal rendelkezik, mint egy Fragment
- A FragmentDialog-ok is rákerülhetnek a BackStack-re

DialogFragment II.

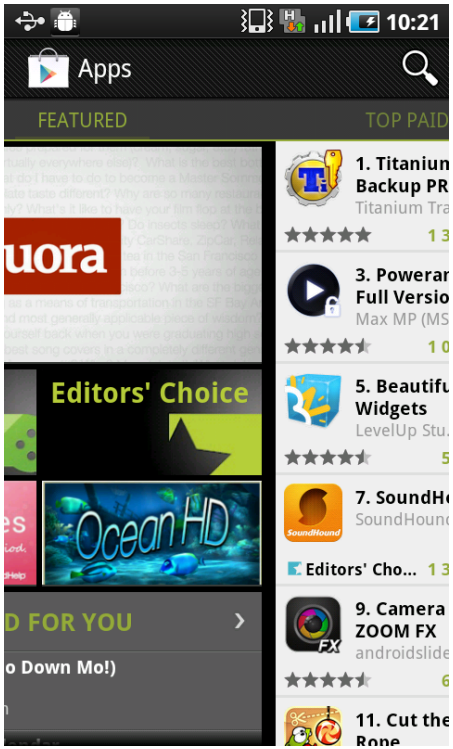
- .onCreateDialog()
  - Ez a metódus is visszatérhet a megjelenítendő Dialog-gal
- .onCreateView()
  - Ha nem használjuk az .onCreateDialog()-ot
  - Tetszőleges megjeleníthető tartalom
- Egy DialogFragment egyben Fragment is!
  - Ha kell, akár Activity-be ágyazottan is megjeleníthető

Fragment-ek fej nélkül

- Ha nem adjuk hozzá a Fragmentet egy ViewGroup-hoz -> a háttérben dolgozik
- .setOnRetainInstanceState(true)
  - Az .onRetainNonConfigurationInstance() helyett használható
  - Forgatásnál csak lecsatolja a megsemmisülő Activity-ről a Fragmentet, majd rácsatolja az újra
    - A memóriában marad, nincs szükség állapotmentésre

ViewPager

- ViewGroup, ahol az elemek közt swipe-al lehet mozogni
  - Pl.: Google Play



FragmentPagerAdapter I.

- Általában Fragment-eket adják a ViewPager oldalait
- Ekkor egy FragmentPagerAdapter példány szolgáltatja az egyes oldalakat
  - Hasonló elven működik, mint a BaseAdapter
  - Fragment getItem(int position):
    - Visszaadjuk a megfelelő Fragment példányt
  - int getCount():
    - Visszaadjuk, hogy összesen hány oldalunk van

FragmentPagerAdapter II.

- String getTitle(int position):
  - Visszaadjuk az adott oldal címét

- Demo!

PagerTitleStrip

- Widget, ami mutatja a ViewPager-ben szereplő oldalak címeit
  - A Support lib. tartalmazza
  - Nem interaktív

PagerTabStrip

- Widget, ami mutatja a ViewPager-ben szereplő oldalak címeit
  - A Support lib. tartalmazza
  - Interaktív!

ViewFlipper komponens

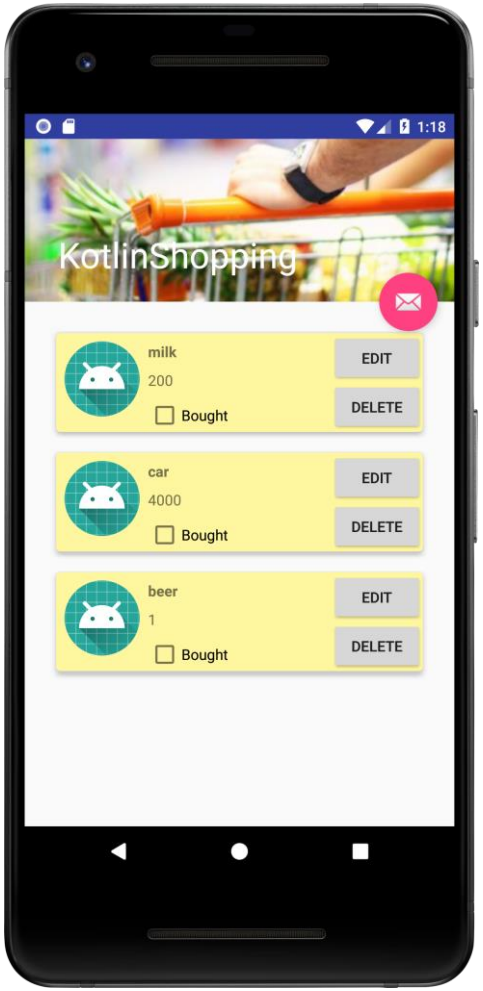
- View közti egyszerű váltás
- Több view található rajta, de mindig csak egyet jelenít meg
- XML-ben felsorolhatók a tartalmazott view-k
- Dinamikusan vezérelhető
- Fragmenteket is kezel
- Ha sok view-t teszünk rá, lassú lehet, mivel mindegyik komponenst előre előkészíti!

További RecyclerView lehetőségek

Gesztusok

Bevásárló lista példa - RecyclerView

- Termékek listázása
  - *Név, ár, vásárlás állapota*
- Új termék felvitele
  - DialogFragment
- Törlés
- Szerkesztés
- Touch gesztusok



Swipe és drag&drop gesztusok

- Távolítsuk el az elemeket swipe hatására
- Tegyük lehetővé az elemek átrendezését drag&drop-pal
- *RecyclerView* támogatás:
  - `ItemTouchHelper.Callback`

ItemTouchHelper.Callback 1/2

- *isLongPressDragEnabled():*
  - True visszatérés ha a drag&drop támogatott
- *isItemViewSwipeEnabled():*
  - True visszatérésé ha a swipe támogatott
- *onMove(...):*
  - Elem mozgatás esetén hívódik meg
- *onSwipe(...):*
  - Swipe esetén hívódik meg

ItemTouchHelper.Callback 2/2

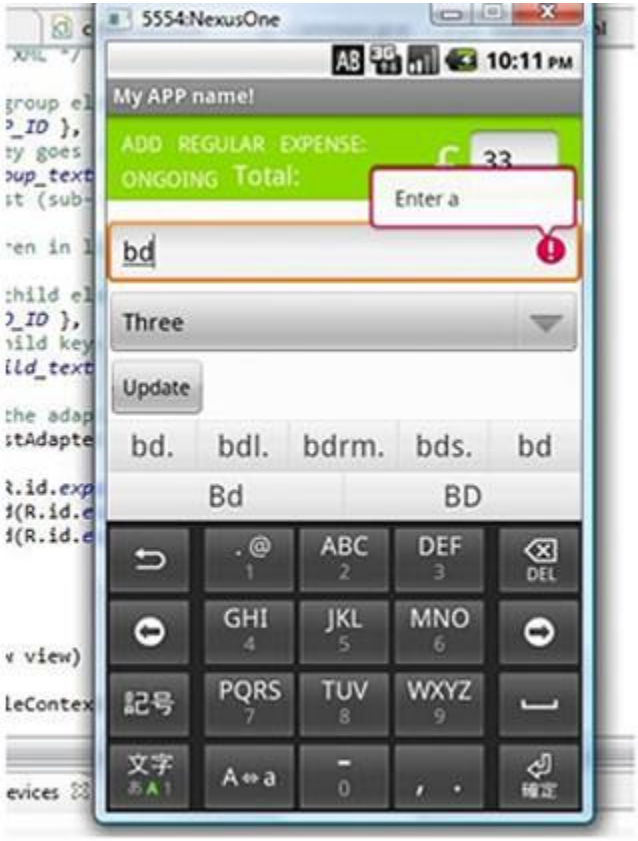
- Drag és swipe irányok beállítása:

```
override fun getMovementFlags(recyclerView: RecyclerView,
  viewHolder: RecyclerView.ViewHolder): Int {
    val dragFlags = ItemTouchHelper.UP or ItemTouchHelper.DOWN
    val swipeFlags = ItemTouchHelper.START or ItemTouchHelper.END
    return ItemTouchHelper.Callback.makeMovementFlags(dragFlags, swipeFlags)
}
```

További UI lehetőségek

Validáció támogatása

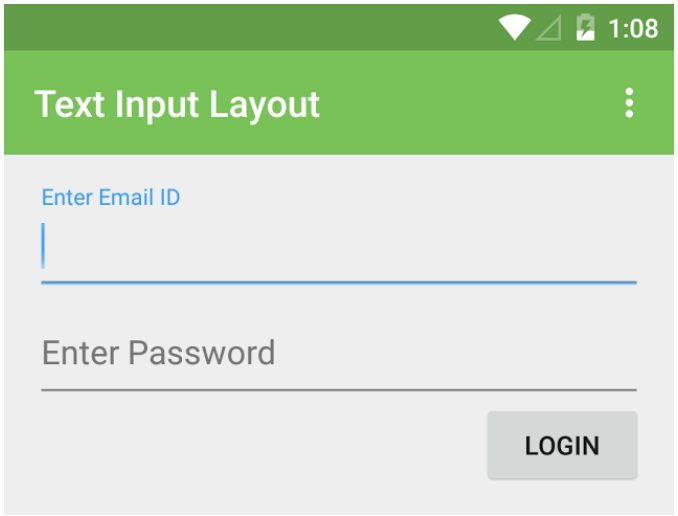
- *setError(errorText)* (pl. EditText-nél)



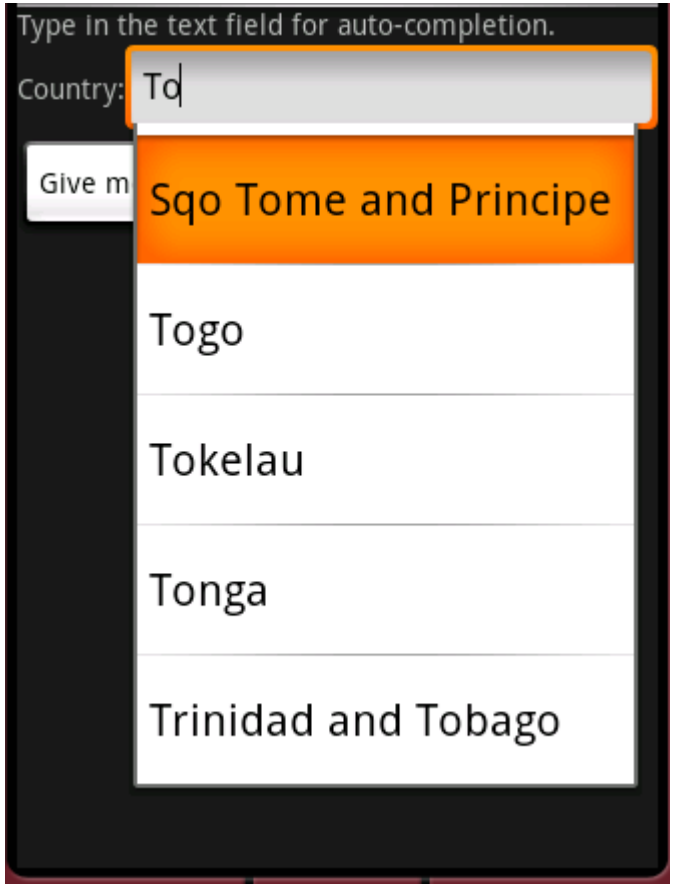
TextInputLayout

- A *Design Support Library* része
- Gradle dependency:
  - *compile 'com.android.support:design:22.2.0'*
- Használat:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter Email ID"
      android:id="@+id/etName" />

</android.support.design.widget.TextInputLayout>
```



Tipikusan Google-s: Autocomplete 1/3



Tipikusan Google-s: Autocomplete 2/3

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" >
        <requestFocus />
    </AutoCompleteTextView>
</LinearLayout>
```

Tipikusan Google-s: Autocomplete 3/3

```
var cityNames = arrayOf("Budapest", "Bukarest", "Krakkó", "Bécs")


override fun onCreate(savedInstanceState: Bundle) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.main)


  var cityAdapter: ArrayAdapter<String> =
  ArrayAdapter<String>(this,
      android.R.layout.
        simple_dropdown_item_1line, cityNames)
  autoCompleteTextView1.setAdapter(cityAdapter)
}
```

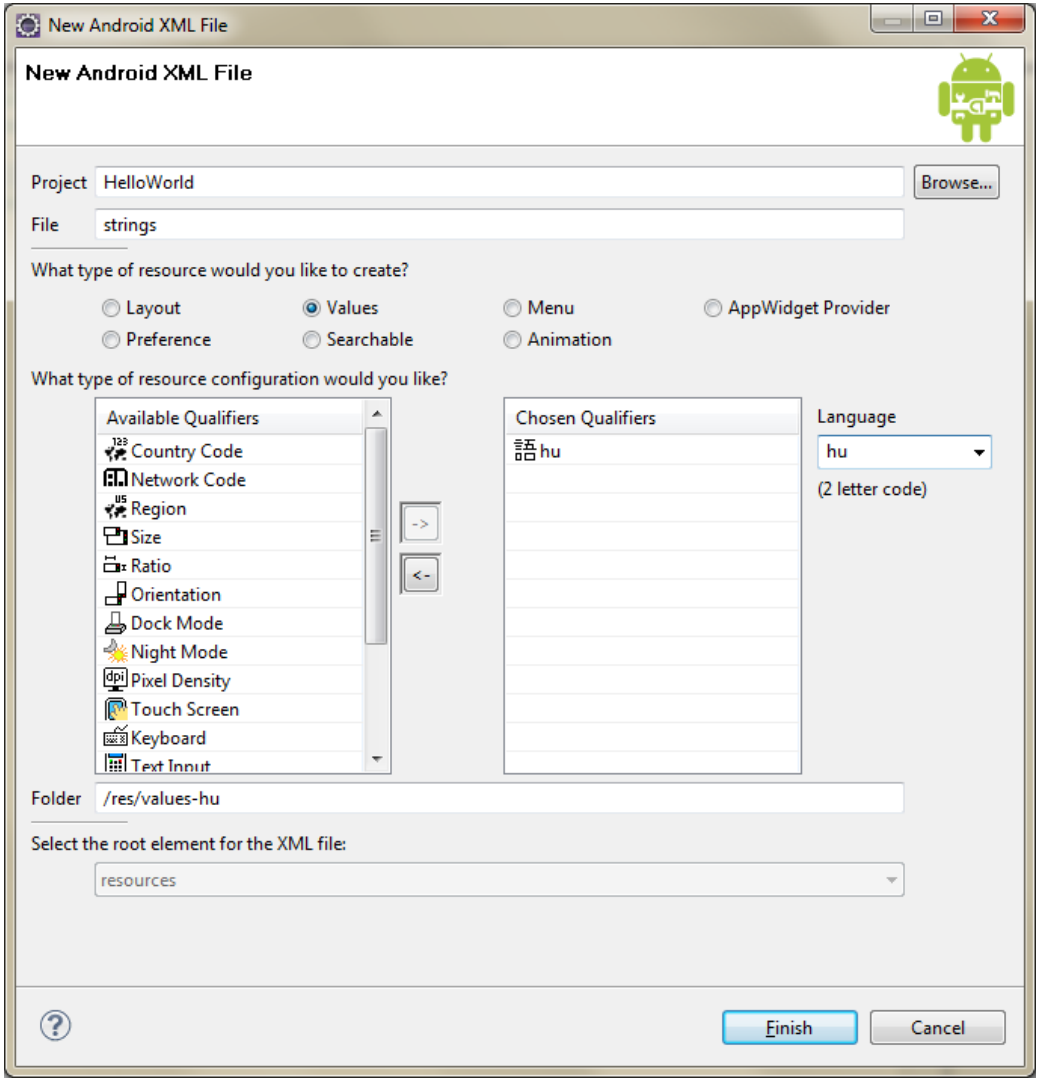
Szöveges erőforrások

- res/values/strings.xml
- Többnyelvűség
- Paraméterezhetőség:
  - <string name="timeFormat">%1$d minutes ago</string>
  - Használat:
    - Context.getString(R.strings.timeFormat, 14)

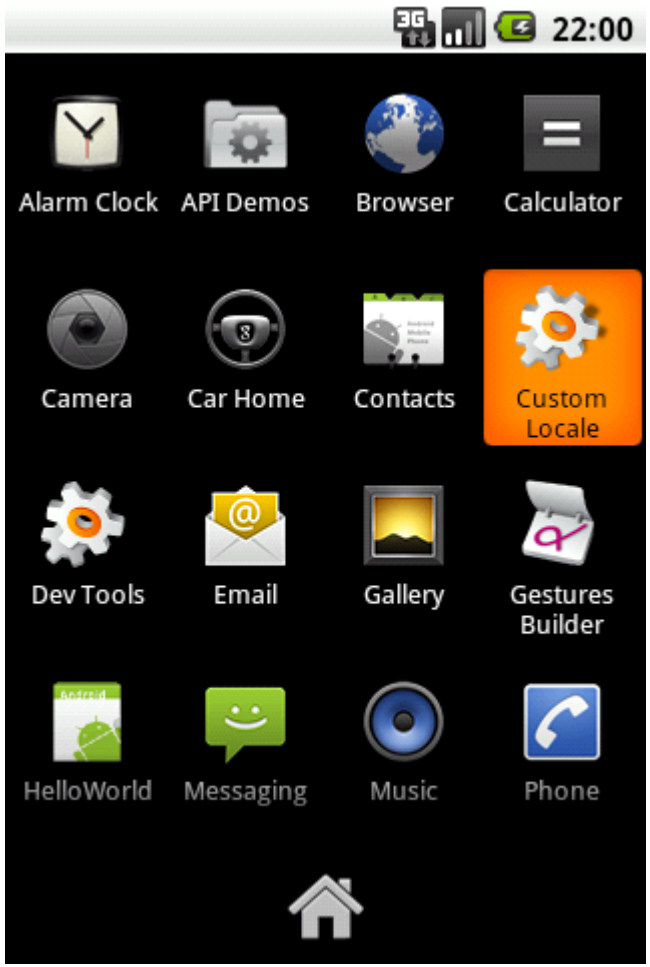
Lokalizáció

- Többnyelvűség támogatása
- Nyelvfüggő felület és erőforrások
- Lokalizációt támogató erőforrás típusok:
  - Képek
  - Elrendezések
  - Szöveges erőforrások
- Alapértelmezett könyvtárak: itt keres a rendszer, ha nincs a kiválasztott lokalizációnak megfelelő erőforrás
  - *res/drawable*
  - *res/layout*
  - *res/value*

Lokalizált erőforrás hozzáadása



Lokalizáció tesztelése Emulátoron



Egyedi nézetek

- View leszármazott
- Beépített nézetek és *LayoutGroup*-ok is felüldefiniálhatók, pl. saját nézet *RelativeLayout*-ból leszármaztatva
- *<merge>* XML elem
- XML-ek egymásba ágyazhatósága: *<include>*

Egyedi felületi nézet

- Teljesen egyedi felületi elemek definiálása
- Meglévő felületi elemek kiegészítése
- Érintés események kezelése
- Dinamikus rajzolás
  - Színek, rajzolási stílus
  - Gyakori alakzatok: vonal, négyzet, kör stb.
  - Szöveg rajzolása
  - Képek megjelenítése
- Megjelenítési méretthez való igazodás
- XML-ből is használható!

CustomView feladatai

- Törekedjünk az Android szabványok betartására
- Támogassuk a szükséges attribútumok XML-ben való megadását is
- Eseménykezelés, paraméterekhez való hozzáférés biztosítása
- Képernyőméret és eszköz függetlenség biztosítása

CustomView példa

- Forráskód:

```
class MyView(context: Context?, attrs: AttributeSet?) :
    View(context, attrs)
```

- XML:

```
<hu.bme.aut.amorg.examples.customview.MyView
    android:id="@+id/myView"
    android:layout_width="fill_parent"
    android:layout_height="100dp"/>
```

XML tulajdonságok megadása

- values/attrs.xml:

```
<resources>
    <declare-styleable name="MyView">
        <attr name="myBgColor" format="color" />
    </declare-styleable>
</resources>
```

- Namespace jelölés (pl. gyökér layout elemben):

```
xmlns:custom="http://schemas.android.com/apk/res/hu.bme.aut.amorg.examples.customview"
```

Paraméter beállítása:

```
<hu.bme.aut.amorg.examples.customview.MyView
    ...
    custom:myBgColor="#ff00ff00" />
```

XML tulajdonságok lekérdezése

```
class MyView(ctx: Context, attrs: AttributeSet) : View(ctx, attrs) {
    private val bgPaint: Paint = Paint()

    init {
        bgPaint.setStyle(Paint.Style.FILL)
        val a = ctx.getTheme().obtainStyledAttributes(
                attrs, R.styleable.MyView, 0, 0)
        try {
            bgPaint.setColor(a.getColor(
                    R.styleable.MyView_myBgColor,
                    Color.RED))
        } finally {
            a.recycle()
        }
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        canvas.drawRect(30F, 30F, width.toFloat() - 30,
          height.toFloat() - 30, bgPaint)
    }
}
```

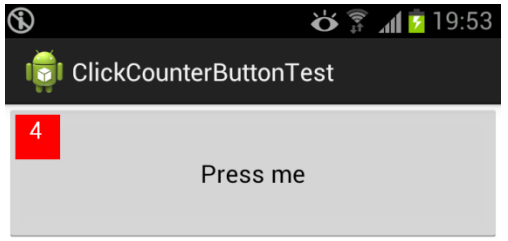
Méret beállítása

- XML-ben megadható a méret
- Dinamikusan is módosítható
- Például négyzetes megjelenítés szélességhez igazítva:

```
override fun onMeasure(widthMeasureSpec: Int,
                       heightMeasureSpec: Int) {
    val w = View.MeasureSpec.getSize(widthMeasureSpec)
    val h = View.MeasureSpec.getSize(heightMeasureSpec)
    val d = if (w == 0) h else if (h == 0) w else if (w < h) w else h
    setMeasuredDimension(d, d)
}
```

Gyakoroljunk!

- Készítsünk egy alkalmazást, amely egy saját gomb vezérlőt jelenít meg!
- Valósítsuk meg, hogy a gombra kattintáskor egy számláló a gomb bal felső sarkában mutassa hányszor kattintottunk a gombra!
- A számláló mérete és láthatósága legyen XML paraméterből állítható!



ConstraintLayout

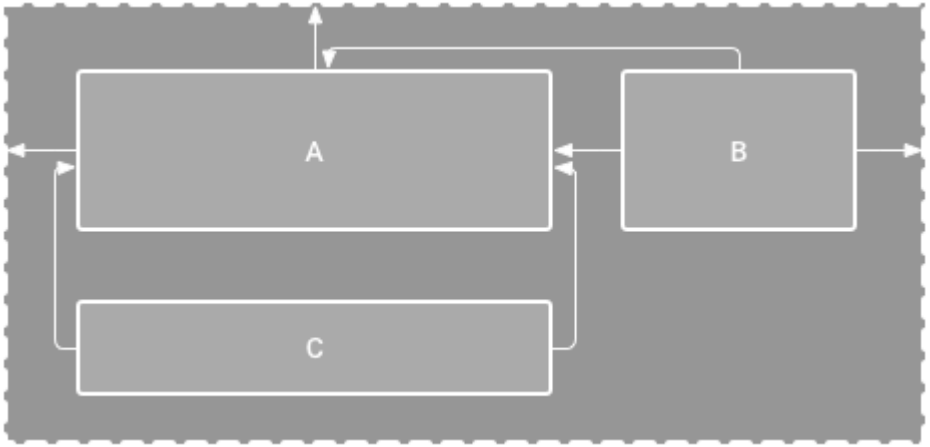
Reszponzív felületek ConstraintLayout-al

- Összetett, komplex layout-ok flat view hierachiával
  - Nincs szükség egymásba ágyazott layout-okra
- RelativeLayout-hoz hasonló
- Layout Editor támogatás
- Támogatás Android 2.3-tól (API Level 9)
- Komplex példák:
  - https://github.com/googlesamples/android-ConstraintLayoutExamples

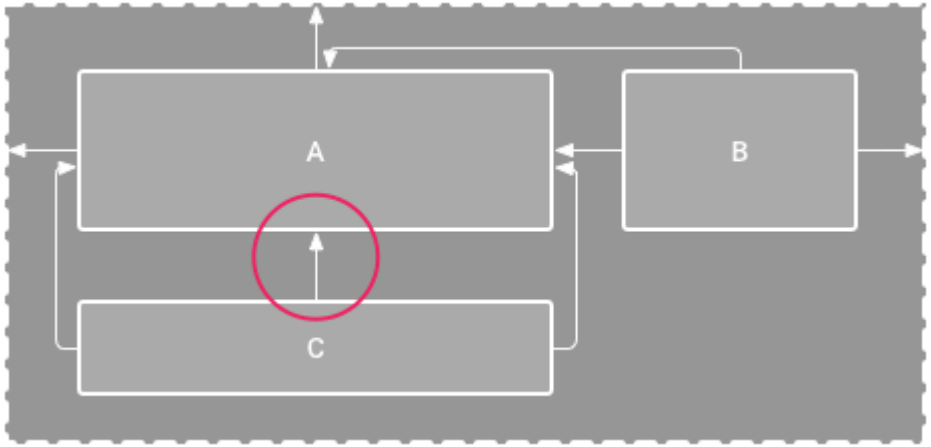
Áttekintés

- Pozíció megadáshoz szükséges:
  - Horizontális és vertikális „szabály” (constraint)
- Minden szabály egy kapcsolat (connection)/igazítás (alignment):
  - Egy másik view-hez képest
  - Szülőhöz képest
  - Egy láthatatlan sorvezetőhöz (guideline) képest
- Attól még, hogy a *LayoutEditor*-ban jól néz ki, nem biztos, hogy eszközön is jó lesz
- Android Studio jelzi a hiányzó szabályokat

Hibás:

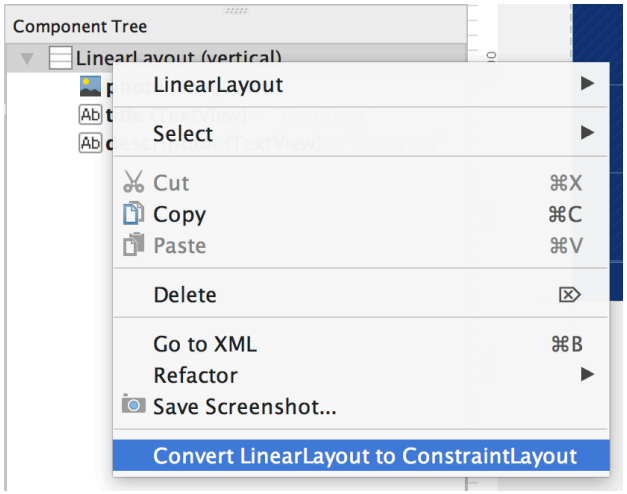


Helyes, mert C tudja, hogy A alatt van:



ConstraintLayout eszközök

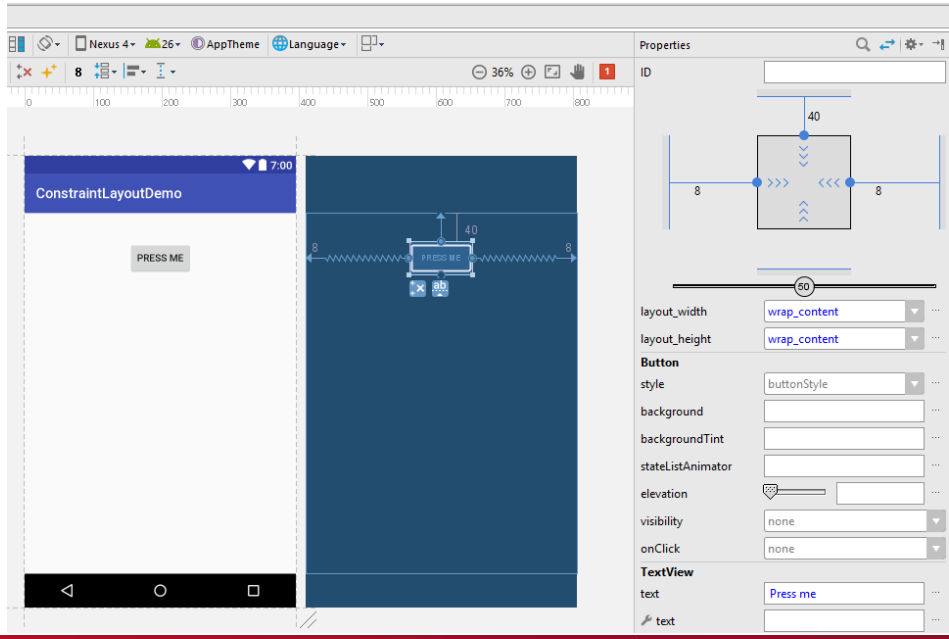
- Gradle import:
  - compile 'com.android.support.constraint:constraint-layout:1.0.2'
- Automatikus átalakítás
  - Nem tökéletes…



ConstraintLayout használat

- Kötelező legalább egy horizontális és vertikális „szabály”

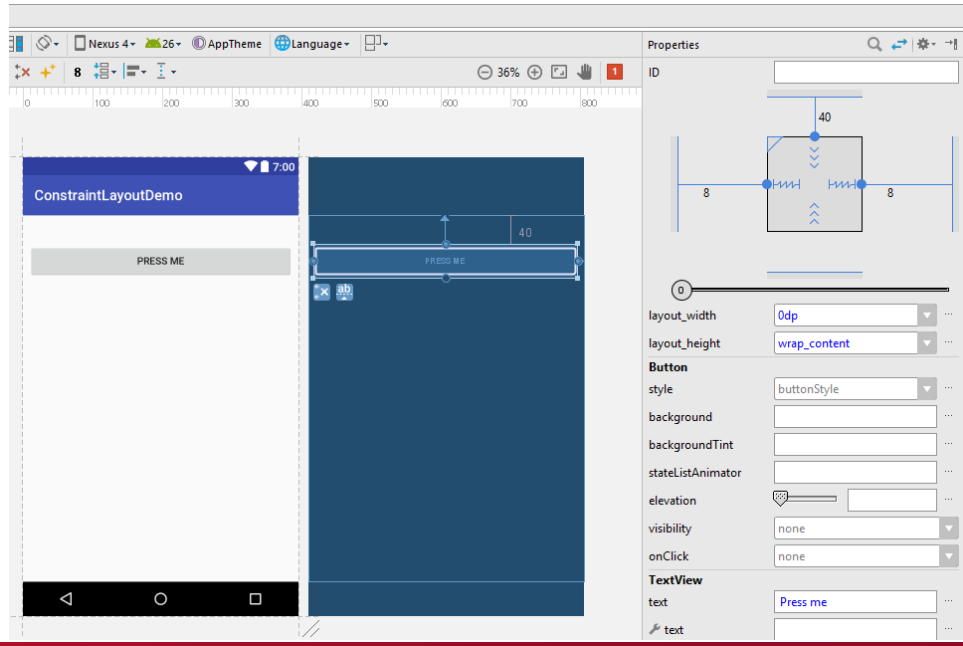
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp„
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```



Mekkora lesz a gomb mérete?

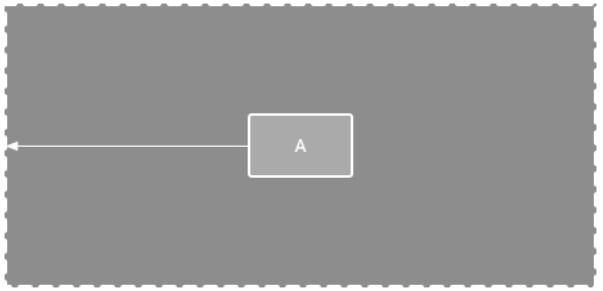
- Nem egyértelmű a szélesség, ellentétes szabályok, jele: 
  - Kettő közé helyezi
- Helyette automata méretezés:
  - **android:layout_width="0dp"**

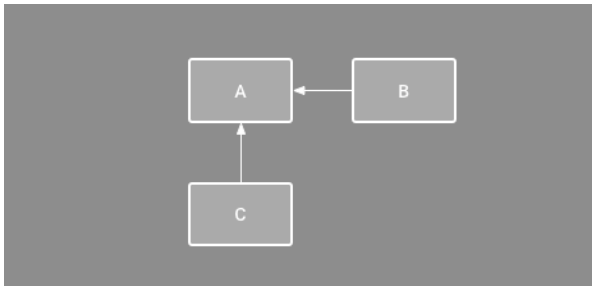
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp„
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```

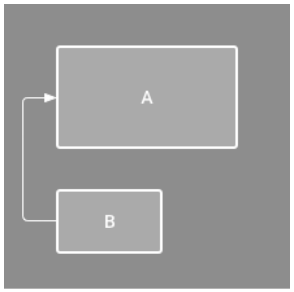


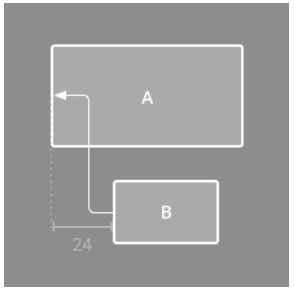
Constraint lehetőségek

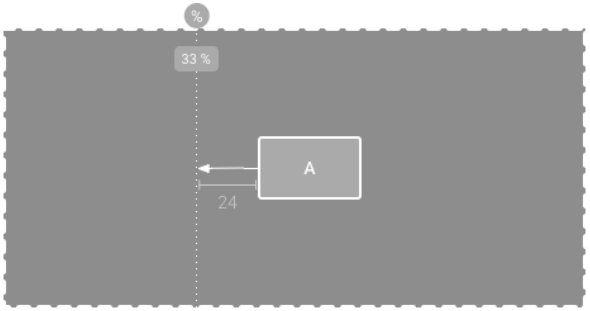
- Szülőhöz képest
- Másik View széleihez képest
- Másik View alapvonalához képest
- Guidelinehez (láthatatlan vezetővonalhoz)

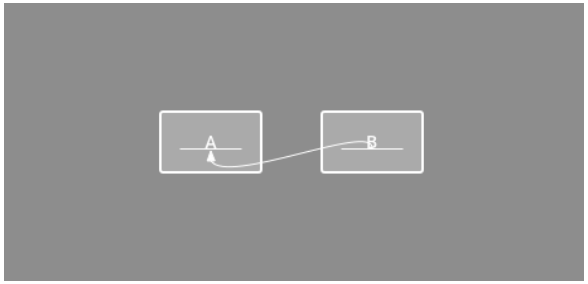












ConstraintLayout teljesítmény

- https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html

Komponens közi kommunikáció

Bevezetés

- A legtöbb platformon az alkalmazások egymástól elkülönítve futnak
  - Minden app a saját „homokozójában” (S*andbox*)
  - 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

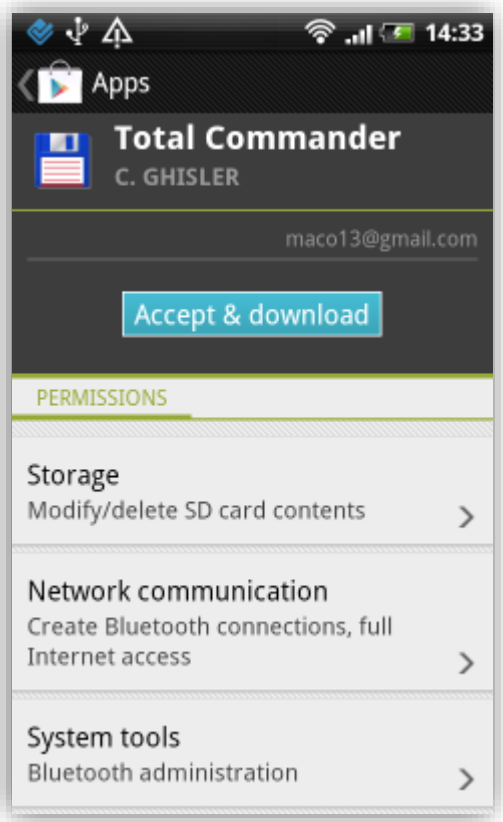
Mi a helyzet Androidon?

- 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

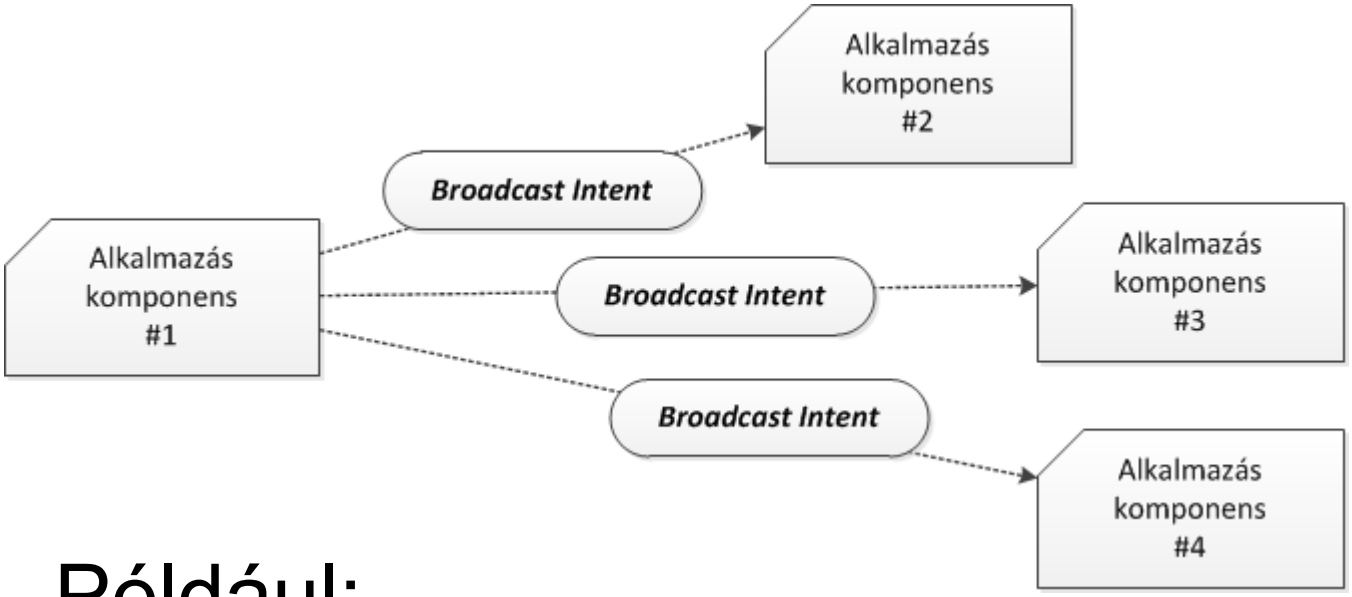
- Egyik komponensből a másikba: ***Intent***

Például:

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

Kommunikáció formái 2/3

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



Például:

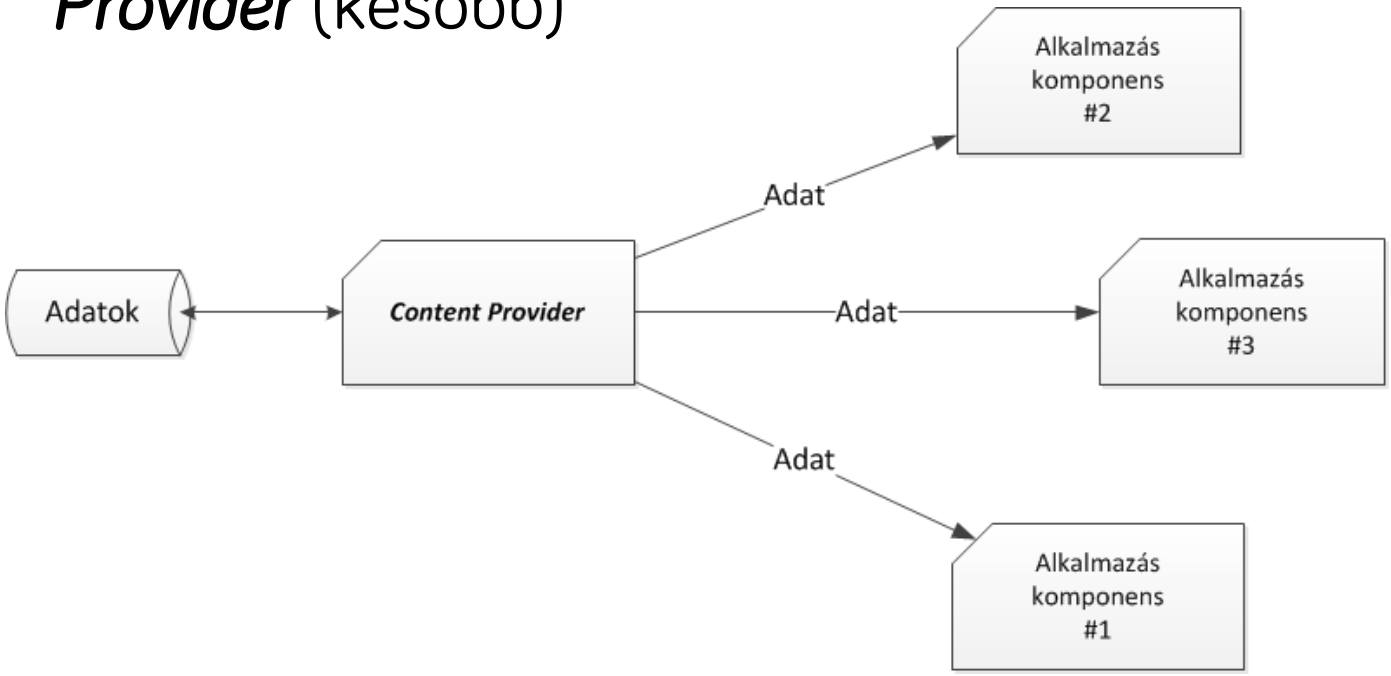
- „Akkufeszültség alacsony” rendszerüzenet

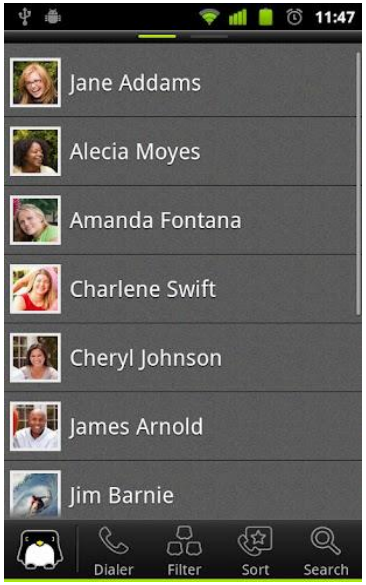
- val intentStart: Intent = Intent(this, „hu.bme.aut.demoapp.DetailActivity”)

- startActivity(intentStart)

Kommunikáció formái 3/3

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



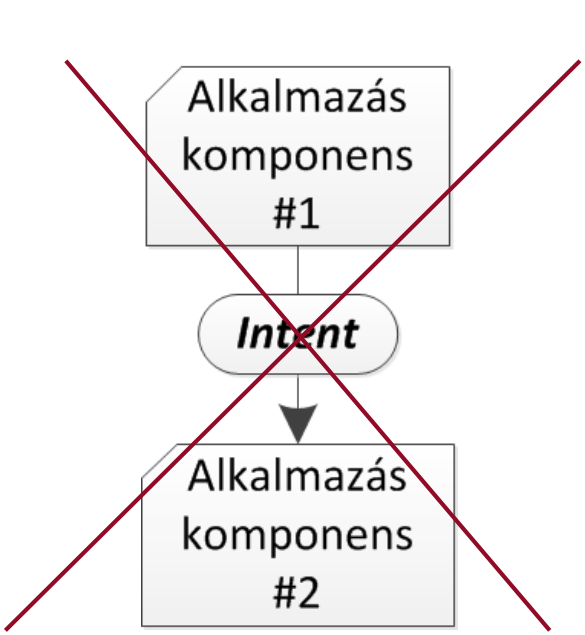


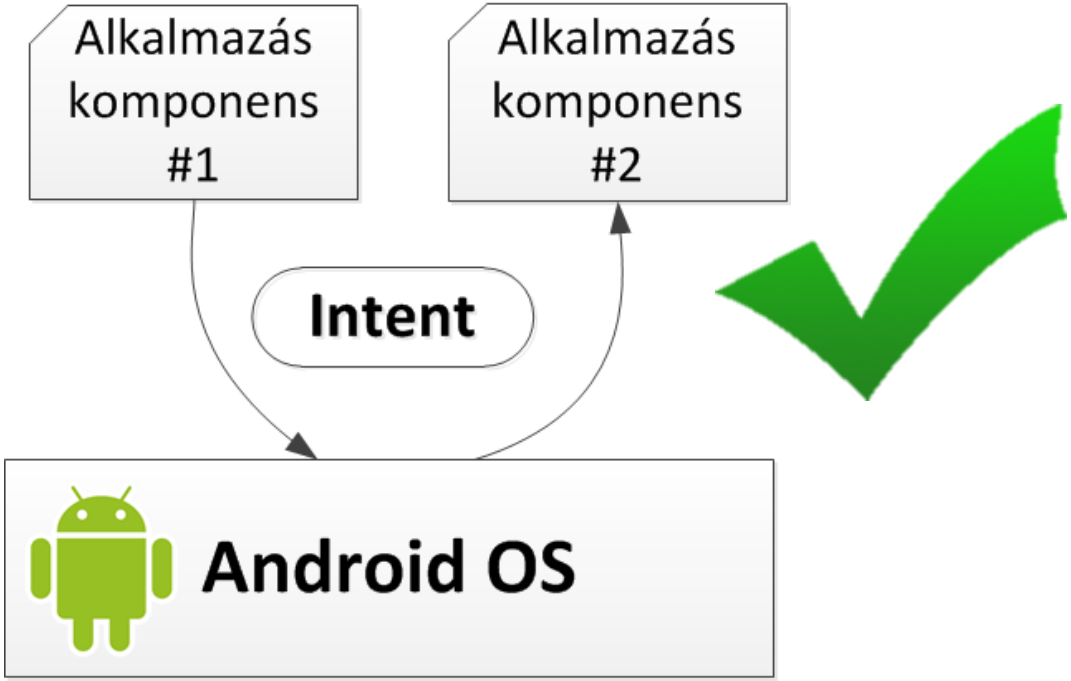
Például:

- Névjegyzék elérése saját alkalmazásból

Intent átadása

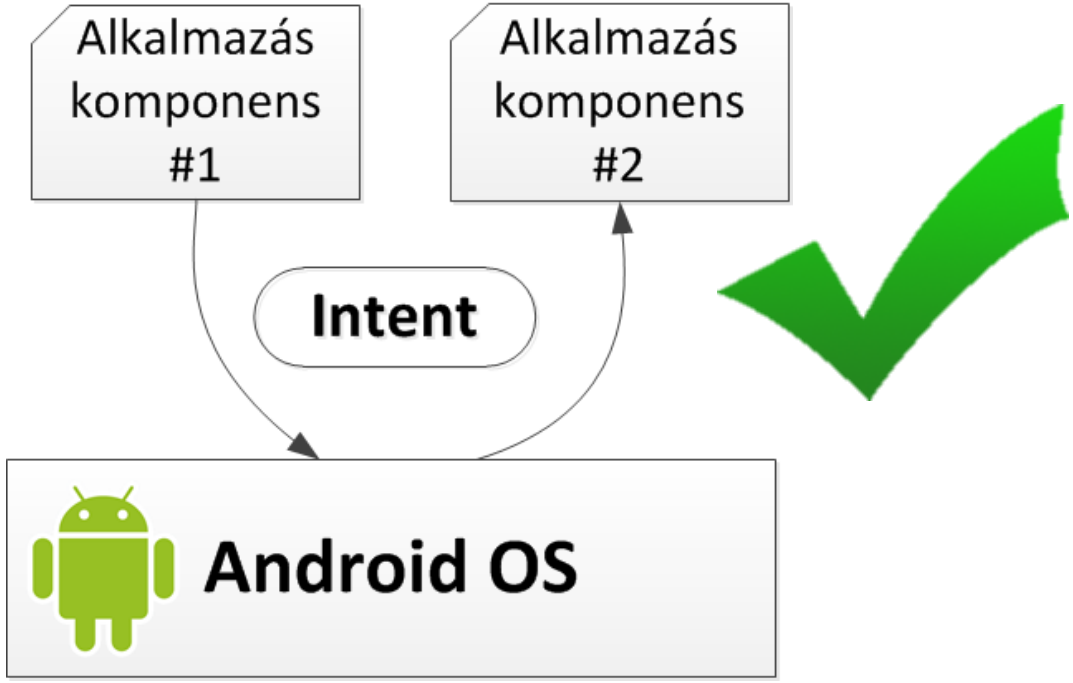
- Mindig az Android runtime-on keresztül!





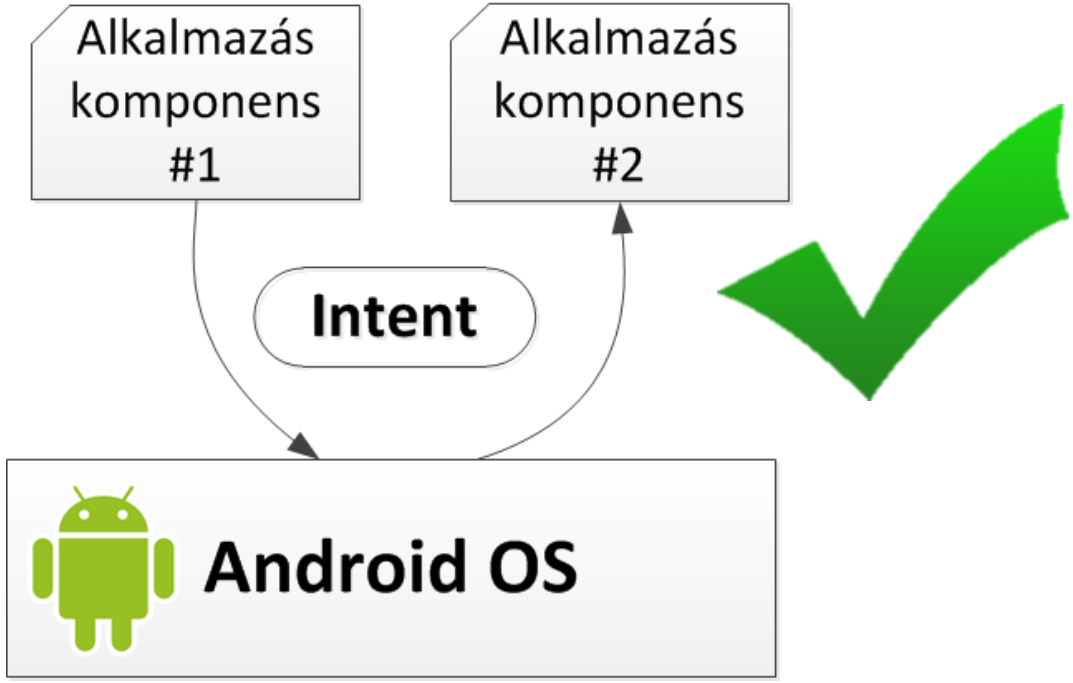
Intent átadása

- Mindig az Android runtime-on keresztül!



Intent átadása

- Mindig az Android runtime-on keresztül!



intent

Intent (*szándék*)

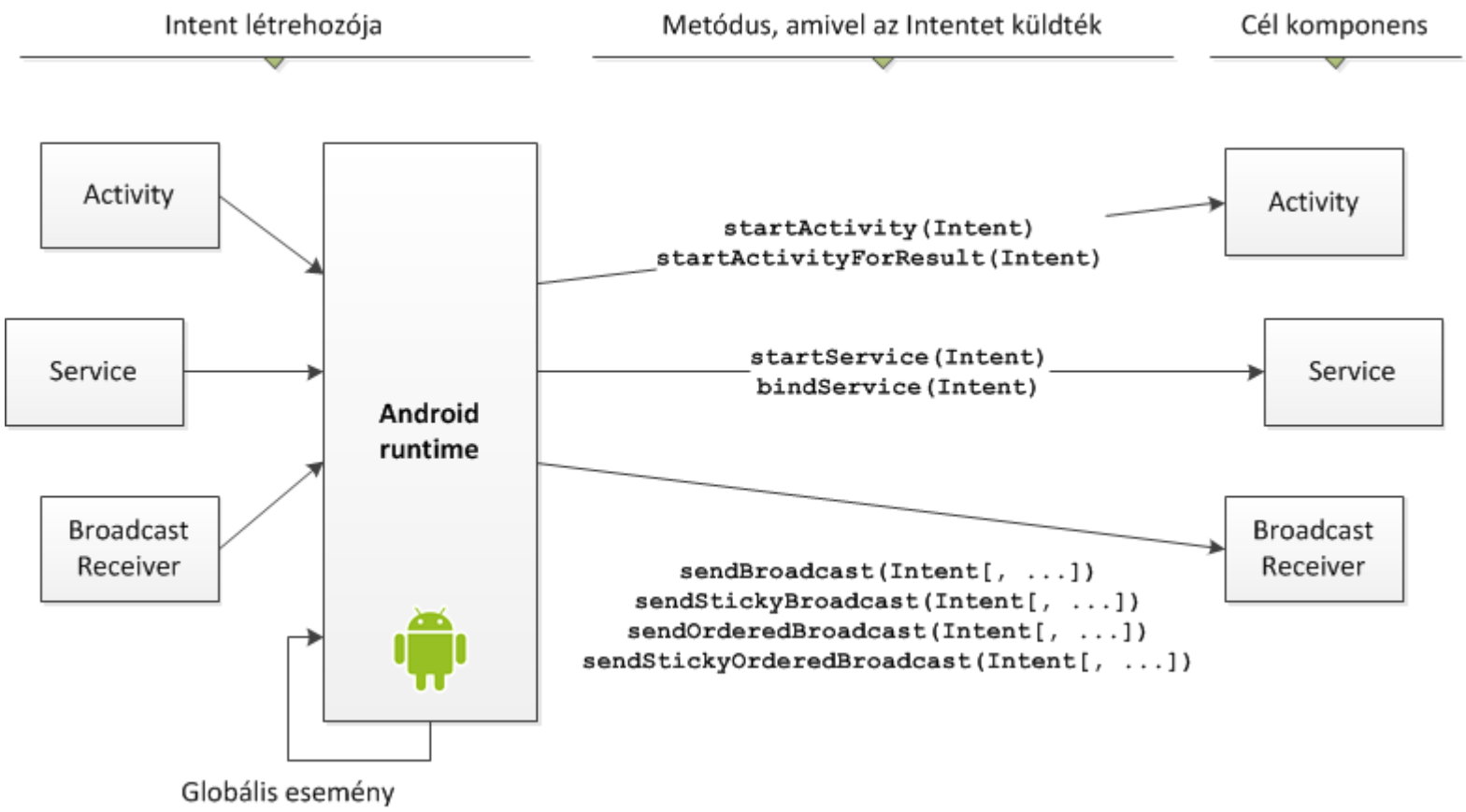
- Passzív adatstruktúra (~struct)
- Késői (futás idejű) kötést valósít meg alkalmazás komponensek között
  - Komponensek: Activity, Service, Broadcast Receiver
- Az elvárt vagy bekövetkezett esemény absztrakt leírása
  - Elvárt esemény leírása, ha az Intent hatására történik valami
    - Activity, Service, 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 létrehozója
Metódus, amivel az Intentet küldték
Cél komponens
Activity
Service
Broadcast Receiver
Android runtime
startActivity(Intent)
startActivityForResult(Intent)
Activity
startService(Intent)
bindService(Intent)
Service
Broadcast Receiver
sendBroadcast(Intent[, ...])
sendStickyBroadcast(Intent[, ...])
sendOrderedBroadcast(Intent[, ...])
sendStickyOrderedBroadcast(Intent[, ...])
Globális esemény

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


Címzett komponens osztályneve
Kategória
Akció
Adat
Extrák
Kapcsolók

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

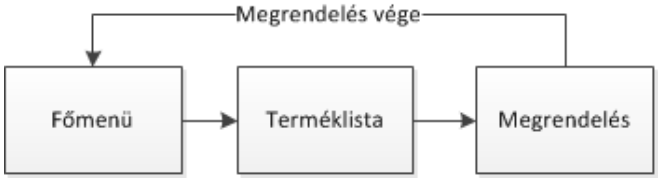
- Mindkét esetben a *startActivity()* függvényt használjuk:
  - **`startActivity(Intent)`**
- **Explicit hívás**: az Intent-ben kitöltjük a címzett komponens nevét (konstruktorból vagy setterrel)

```
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?

startActivityForResult

- **`startActivity()`**-vel indítva nem kapunk visszajelzést
- Megoldás:

  `startActivityForResult(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(resultCode)`** a hívott oldalon

onActivityResult

- Visszatérési érték kezelése a hívó oldalon (callback):

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

- Paraméterei:
  - **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 Parceable interfészt

Intent Extras

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

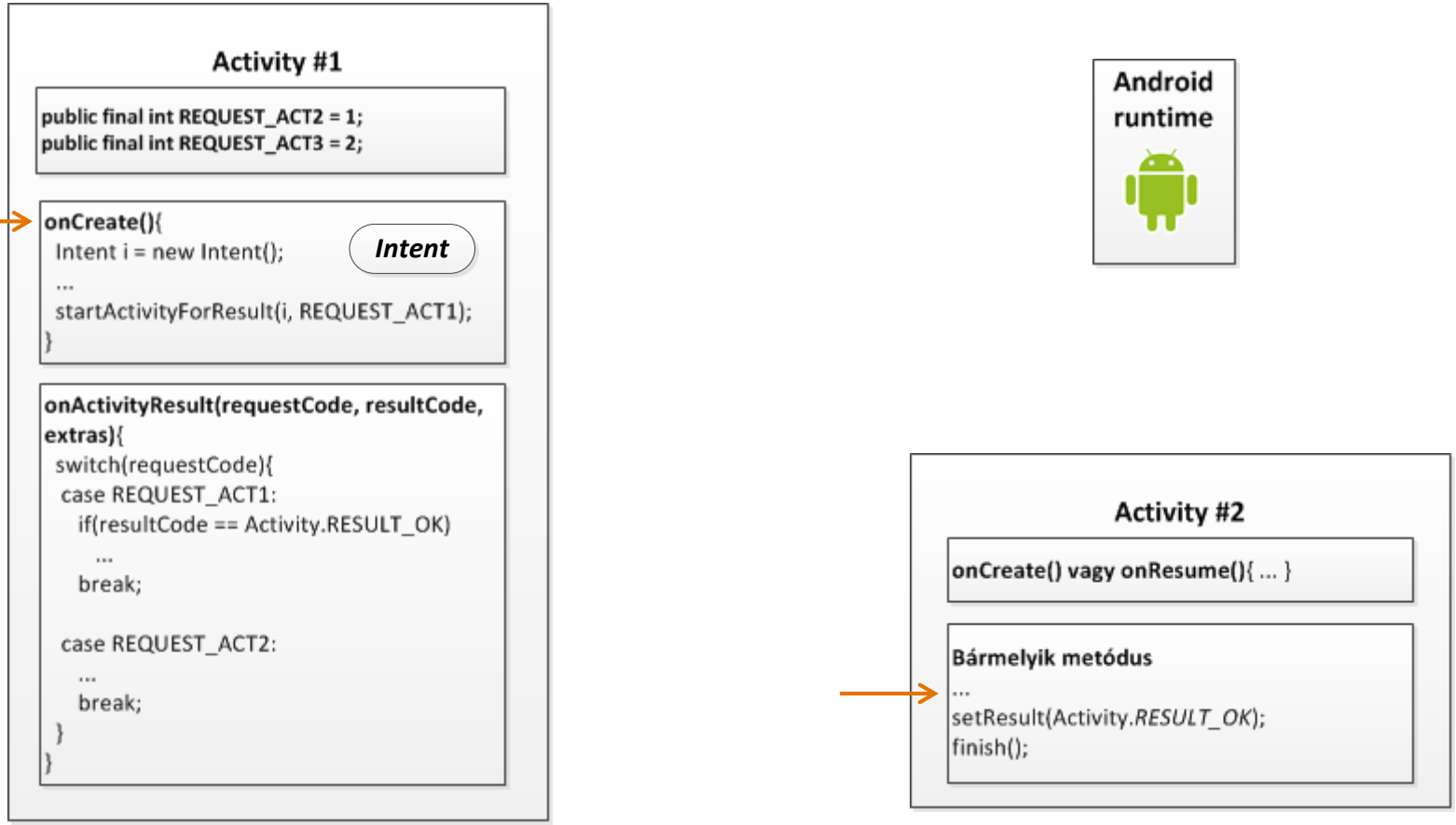
```
var resultIntent: Intent = Intent()
resultIntent.putExtra(
 "UserID", currentUser.getId())
resultIntent.putExtra("role", currentUser.getRole())
setResult(RESULT_OK, resultIntent)
finish()
```

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)
```

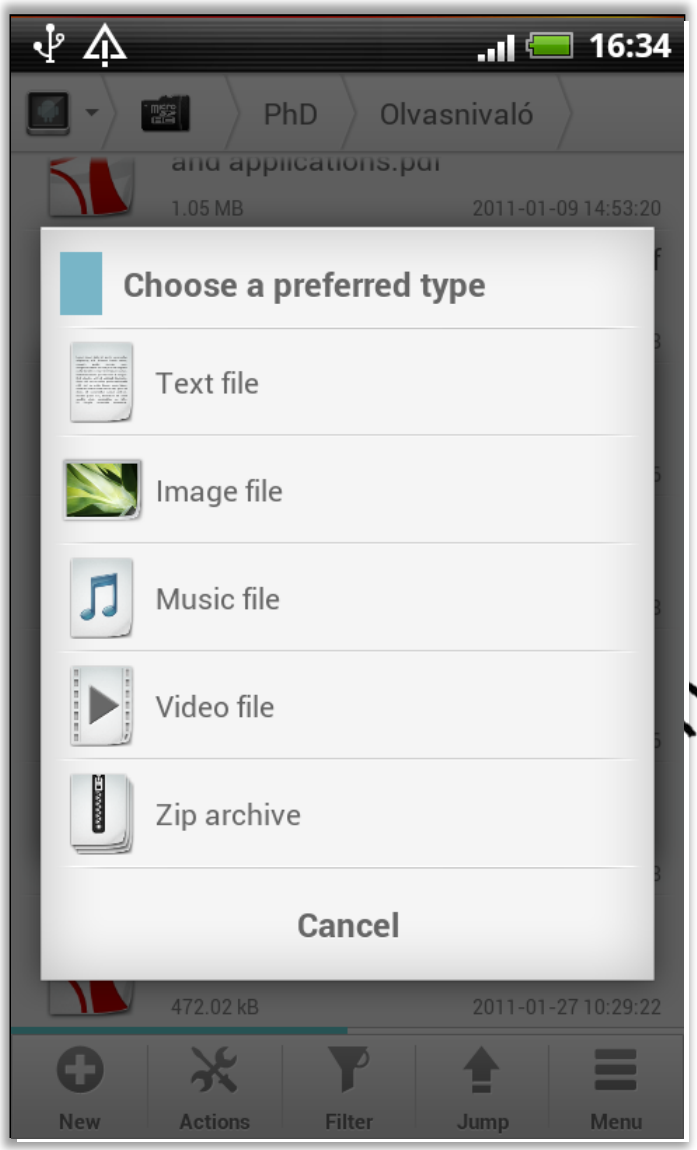
- Névjegy kiválasztása


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

Implicit Intent - Példa

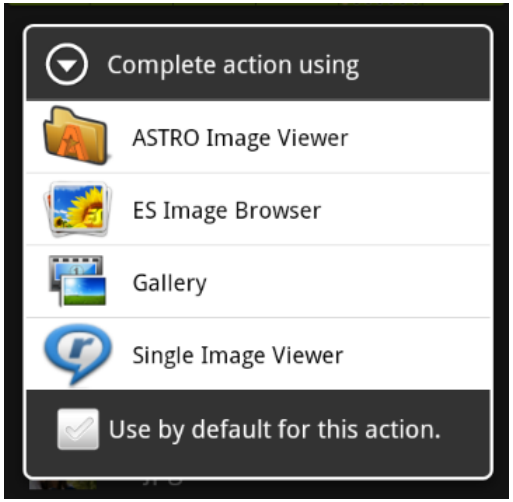
- Pdf megnyitása

```
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)
```


16:34
PhD
Olvasnivaló
and applications.pdf
1.05 MB
2011-01-09 14:53:20
Choose a preferred type
Text file
Image file
Music file
Video file
Zip archive
Cancel
472.02 kB
2011-01-27 10:29:22
New
Actions
Filter
Jump
Menu

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 kiajánlott 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: Boundle = 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
  - Kiajánlhatjuk a sajátunkat

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

Kérdések