

Android fejlesztés

Alkalmazás komponensek, felhasználói felület lehetőségek

peter.ekler@aut.bme.hu



Department of
Automation and
Applied Informatics

Miről volt szó az előző alkalommal? 😊

- Activity komponens
- Activity élelciklus
- Több Activity-s alkalmazások
- Állapot megőrzés az élelciklus során
- Menük kezelése
- Felhasználói felület tervezése, egyedi nézetek, rajzolás

Android LogCat

- Rendszer debug kimenet
- Beépített rendszer üzenetek is monitorozhatók
- Beépített Log osztály
 - > v(String, String) (verbose)
 - > d(String, String) (debug)
 - > i(String, String) (information)
 - > w(String, String) (warning)
 - > e(String, String) (error)
- Log.i("MyActivity", "Pozíció: " + position);
- Átirányítható file-ba is
 - > *adb logcat > textfile.txt*

Tartalom

- Kotlin emlékeztető
- Alkalmazás komponensek
- Felhasználói felület tervezés és eszközök
- Layout-ok
 - > LinearLayout
 - > RelativeLayout
 - > ConstraintLayout
- Egyszerű View elemek

Kotlin alapok

Forrás: <https://kotlinlang.org/docs/reference/>



A Kotlin főbb jellemzői

- JVM byte kódra (vagy akár JavaScriptre is) fordul
- Meglévő Java API-k, keretrendszerek és könyvtárak használhatók
- Automatikus konverzió Java-ról Kotlinra
- Null-safety
 - > Vége a NullPointerException korszaknak
- Kód review továbbra is egyszerű
 - > A nyelv alapos ismerete nélkül is olvasható a kód

Konstansok, változók (val vs. var)

- Egyszeri értékadás – „val”

```
val score: Int = 1 // azonnali értékadás
val idx = 2      // típus elhagyható
val age: Int     // típus szükséges ha nincs azonnali értékadás
age = 3          // későbbi értékadás
```

- Változók (megváltoztatható) – „var”

```
var score = 0 // típus elhagyható
score += 1
```

- String sablonok

```
var score = 1
val scoreText = "$score pont"
```

```
score = 2
// egyszerű kifejezések string-ek esetében:
val newScoreText = "${scoreText.replace("pont", "volt, most ")} $score"
```

Változók null értéke

- Alapból a változók értéke nem lehet null

```
var a: Int = null  
error: null can not be a value of a non-null type Int
```

- A '?' operátorral engedélyezhetjük a null értéket

```
var a: Int? = null
```

```
var x: List<String?> =  
  listOf(null, null,  
    null)
```

> Lista, melyben lehetnek null elemek

> Lista, mely lehet null

```
var x: List<String>? = null
```

> Lista, mely lehet null és az
elemei is lehetnek null-ok

```
var x: List<String?>?  
= null  
x = listOf(null,  
  null, null)
```


Null tesztelés és az Elvis operátor

```
var nullTest : Int? = null  
nullTest?.inc()
```

> `inc()` nem hívódik meg, ha `nullTest` `null`

```
var x: Int? = 4  
var y = x?.toString() ?: ""
```

> ha `x` `null`, akkor `y` `""` értéket kap

“Double bang” operator

- Kivételt dob, ha a változó értéke null

```
var x: Int? = null  
x!!.toString()  
kotlin.KotlinNullPointerException
```

Függvények

- Függvény szintaxis

```
fun add(a: Int, b: Int): Int {  
    return a + b  
}
```

- Kifejezés törzs, visszatérési típus elhagyható

```
fun add(a: Int, b: Int) = a + b
```

- Érték nélküli visszatérés – Unit

```
fun printAddResult(a: Int, b: Int): Unit {  
    println("$a + $b értéke: ${a + b}")  
}
```

- Unit elhagyható

```
fun printAddResult(a: Int, b: Int) {  
    println("$a + $b értéke: ${a + b}")  
}
```

Osztályok

```
class Car {  
    private String type;  
    public Car(String type) {  
        this.type = type;  
    }  
}  
  
class Car constructor(val type: String) {  
    val typeUpper = type.toUpperCase()  
  
    init {  
        Log.d("TAG_DEMO", "Car created: ${type}")  
    }  
  
    constructor(type: String, model: String) : this(type) {  
        Log.d("TAG_DEMO", "Car model: ${model}")  
    }  
}
```

Osztályok

```
class Car constructor(val type: String) {  
    val typeUpper = type.toUpperCase()  
  
    init {  
        Log.d("TAG_DEMO", "Car created: ${type}")  
    }  
  
    constructor(type: String, model: String) : this(type) {  
        Log.d("TAG_DEMO", "Car model: ${model}")  
    }  
}
```

constructor elhagyható

primary constructor
paraméterekkel

primary constructor
tagváltozóira lehet
hivatkozni

primary constructor
inicializáló blokk

secondary constructor

```
// példányosítás  
val car = Car("Toyota")
```

Leszármaztatás

alap esetben minden
final

```
open class Item(price: Int) {  
    open fun calculatePrice() {}  
    fun load() {}  
}
```

öröklés

```
class SpecialItem(price : Int) : Item(price) {  
    final override fun calculatePrice() {}  
}
```

Később már nem
lehet felülírni

Osztály elemek

opcionális hozzáférés
módosító

konstruktor
opcionális
hozzáférés
módosítója

kulcsszó (kötelező, ha
van hozzáférés
módosítója a
konstruktornak)

fejléc

konstruktor paraméterek

```
public class Car internal constructor(aPlateNumber: String) {
```

```
    val plateNumber: String
```

```
    var motorNumber: String? = null
```

```
    init {
```

```
        plateNumber = aPlateNumber.toUpperCase();
```

```
    constructor(aPlateNumber: String, aMotorNumber: String):
```

```
        this(aPlateNumber) {
```

```
            motorNumber = aMotorNumber.toUpperCase()
```

```
        }
```

```
    fun start(targetVelocity: Int) {
```

```
        // some code
```

```
    }
```

```
}
```

az elsődleges
konstruktornak
nincs body-ja

inicializáló blokk

másodlagos
konstruktor

függvény

Java field vs. Kotlin property

Java

```
public class Car {  
    private String type;  
  
    public String getType() {  
        return type;  
    }  
  
    public void setType(String type) {  
        Log.d("TAG_CAR", "type SET");  
        this.type = type;  
    }  
}
```

Kotlin

```
class Car {  
    var type: String? = null  
    set(type) {  
        Log.d("TAG_CAR", "type SET")  
        field = type  
    }  
}
```


Alkalmazás komponensek

Android alkalmazás felépítése 1/2

- Egy Android alkalmazás egy vagy több alkalmazás komponensből épül fel:
 - > Activity-k
 - > Service-k
 - > Content Provider-ek
 - > Broadcast Receiver-ek

Android alkalmazás

Activityk

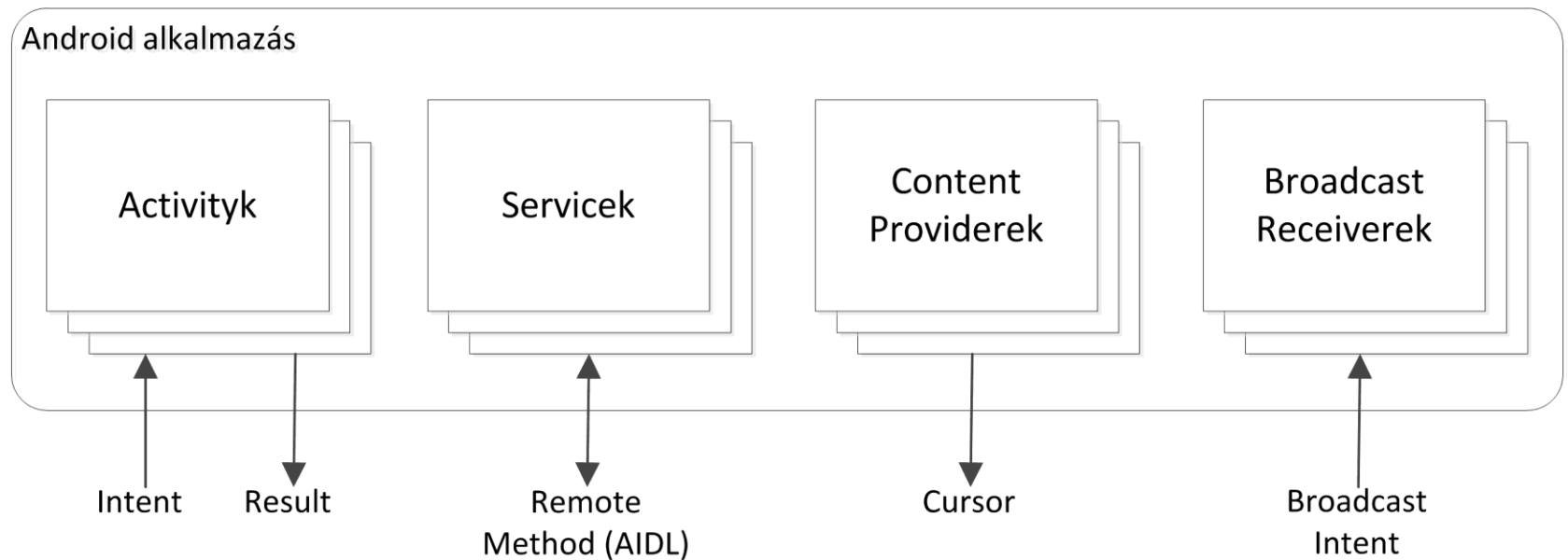
Servicek

Content
Providerek

Broadcast
Receiverek

Android alkalmazás felépítése 2/2

- Minden komponensnek különböző szerepe van az alkalmazáson belül
- Bármelyik komponens önállóan aktiválódhat
- Akár egy másik alkalmazás is aktiválhatja az egyes komponenseket



Activity-k

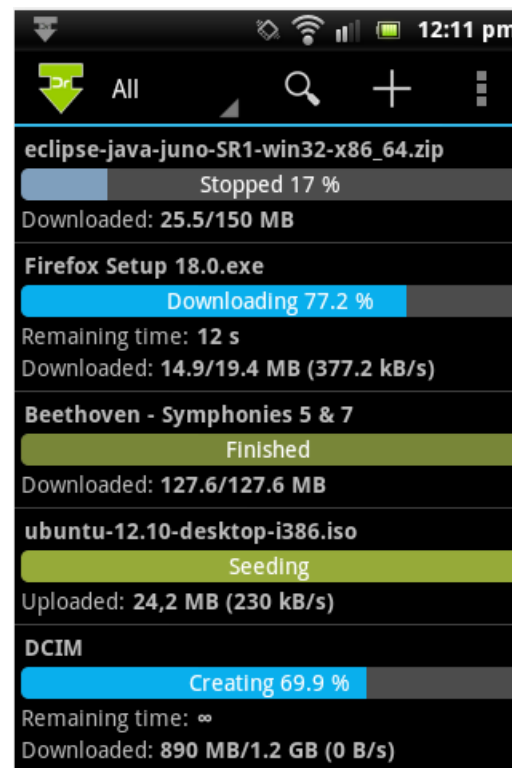
- Különálló nézet, saját UI-al
- Például:
 - > Emlékeztető alkalmazás
 - > 3 Activity: ToDo lista, új ToDo felvitele, ToDo részletek
- Független Activity-k, de együtt alkotják az alkalmazást
- Más alkalmazásból is indítható az Activity, például:
 - > Kamera alkalmazás el tudja indítani az új ToDo felvitele Activity-t és a képet hozzá rendeli az emlékeztetőhöz
- Az **android.app.Activity** osztályból származik le

Service-k

- A Service komponens egy hosszabb ideig háttérben futó feladatot jelképez
- Nincs felhasználói felülete
- Például egy letöltő alkalmazás (torrent 😊) fut a háttérben, míg előtérben egy másik programmal játszunk
- Más komponens (pl. Activity) elindíthatja, vagy csatlakozhat (bind) hozzá vezérlés céljából
- Az `android.app.Service` osztályból kell öröklődnie

DrTorrent

- BitTorrent kliens Android platformra
- Megszokott funkciók és háttérben működés



Content provider-ek

- A Content provider (tartalom szolgáltató) komponens feladata egy megosztott adatforrás kezelése
- Az adat tárolódhat fájlrendszerben, SQLite adatbázisban, web-en, vagy egyéb perzisztens adattárban, amihez az alkalmazás hozzáfér
- A Content provider-en keresztül más alkalmazások hozzáférhetnek az adatokhoz, vagy akár módosíthatják is azokat
- Például: CallLog alkalmazás, ami egy Content provider-t biztosít, és így elérhető a tartalom
- A **android.content.ContentProvider** osztályból származik le és kötelezően felül kell definiálni a szükséges API hívásokat

Broadcast receiver-ek

- A Broadcast receiver komponens a rendszer szintű eseményekre (broadcast) reagál
- Például: kikapcsolt a képernyő, alacsony az akkumulátor töltöttsége, elkészült egy fotó, bejövő hívás, stb.
- Alkalmazás is indíthat saját „broadcast”-ot, például ha jelezni akarja, hogy valamilyen művelettel végzett (letöltődött a torrent 😊)
- Nem rendelkeznek saját felülettel, inkább valamilyen figyelmeztetést írnak ki például a status bar-ra, vagy elindítanak egy másik komponenst (jeleznek például egy service-nek)
- A **android.content.BroadcastReceiver** osztályból származik le; az esemény egy Intent (lásd. Később) formájában érhető el

Hogy is volt?

- Magyarázza el a fordítás mechanizmusát!
- Egy Android alkalmazás milyen komponensekből épülhet fel?
- Mi a Service komponens?
- Miket kell tartalmaznia a manifest állománynak?
- Az Activity callback életciklus-függvények felüldefiniálásakor meg kell-e hívni kötelezően az ősz osztály implementációját?
- Ha A Activity-ből átváltunk B Activity-re, milyen sorrendben hívódnak meg az életciklus függvények?
- Magyarázza el az Activity Back Stack működési elvét!

Felhasználói felület - fogalmak

Különböző képernyők támogatása 1/2

- Az Android futtatható különböző felbontású és sűrűségű képernyőkön
- A rendszer egyfajta mechanizmust biztosít az eltérő képernyők támogatására (1.6-tól felfele)
- A fejlesztő válláról a legtöbb munkát leveszi
- Csak a megfelelő erőforrásokat kell elkészíteni
- Például egy mobiltelefon és egy tablet képernyője tipikusan eltérő
- 3.2-es tablet API-tól felfele újabb módszerek (lásd később)

Különböző képernyők támogatása 2/2

- A rendszer automatikusan is skálázza és átméretezi az alkalmazás felületét, hogy minden készüléket támogasson
- De! mindenképp fontos, hogy a felhasználói felület és az erőforrások (képek) optimalizálva legyenek az egyes felbontásokhoz és sűrűségekhez
- Ezzel nagy mértékben növelhető a felhasználói élmény
- Továbbá valóban az egyes készülékekhez igazítható a megjelenítés, ami növeli a felhasználói elégedettséget
- A módszer követésével minden készüléket támogató alkalmazás készíthető UI szempontjából egyetlen .apk-ba csomagolva

Legfontosabb fogalmak 1/2

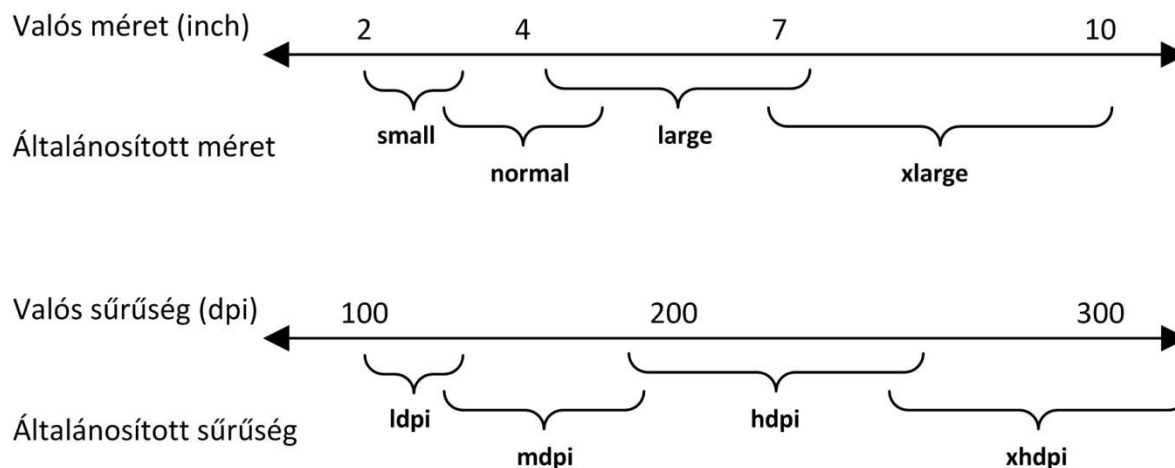
- Képernyő méret (*screen size*):
 - > Fizikai képátló
 - > Az egyszerűség kedvéért az Android 4 kategóriát különböztet meg: small, normal, large, és extra large
- Képernyő sűrűség (*screen density – dpi*): A pixelek száma egy adott fizikai területen belül, tipikusan inchenkénti képpont (dpi – dots per inch)
 - > Az Android 6 kategóriát különböztet meg: low, medium, high és extra high
- Orientáció (*orientation*): A képernyő orientációja a felhasználó nézőpontjából:
 - > Álló (*portrait*)
 - > Fekvő (*landscape*)
 - > Az orientáció futási időben is változhat, például a készülék eldöntésével
 - > Lehetőség van rögzíteni az orientációt

Legfontosabb fogalmak 2/2

- Felbontás (*resolution – px*): Képernyő pixelek száma
 - > A UI tervezésekor nem felbontással dolgozunk, hanem mérettel és pixel sűrűséggel
- Sűrűség független pixel (*density-independent pixel – dp*)
 - > Virtuális pixel egység, amit UI tervezéskor célszerű használni
 - > Egy dp egy fizikai pixelnek felel meg egy 160 dpi-s képernyőn (160 az egységes középérték)
 - > A rendszer futási időben kezel minden szükséges skálázást a definiált dp-nek megfelelően
 - > $px = dp * (dpi / 160)$
 - > Például egy 240 dpi-s képernyőn, 1 dp 1.5 fizikai pixelnek felel meg

Általánosított képernyő méretek 1/2

- 6 általánosított méret:
 - > small, normal, large és xlarge, stb.
- 6 általánosított sűrűség:
 - > ldpi (low), mdpi (medium), hdpi (high), és xhdpi (extra high), stb.



Általánosított képernyő méretek 2/2

- Definiált minimum küszöbök:
 - > *xlarge*: legalább 960dp x 720dp
 - > *large*: legalább 640dp x 480dp
 - > *normal*: legalább 470dp x 320dp
 - > *small*: legalább 426dp x 320dp
- 3.0-ás verzió alatt lehetnek bugok a normal és large megkülönböztetésében

Futás idejű működés

- A megjelenítés optimalizálása érdekében lehetőség van alternatív erőforrások megadására a különböző méretek és sűrűségek támogatásához
- Tipikusan különböző layout-ok és eltérő felbontású képek definiálása szükséges
- A rendszer futási időben kiválasztja a megfelelő erőforrást
- Általában nincs szükség minden méret és sűrűség kombináció megadására

Mi nem igaz az Android UI támogatására?

- A. Az Android automatikusan átméretezi a képet, ha nincs megfelelően illeszkedő.
- B. Az Android támogatja a sűrűségfüggetlen megjelenítést.
- C. $px = dp * (dpi / 160)$
- D. Közvetlenül pixelben nem adhatók meg a méretek.

Mi nem igaz az Android UI támogatására?

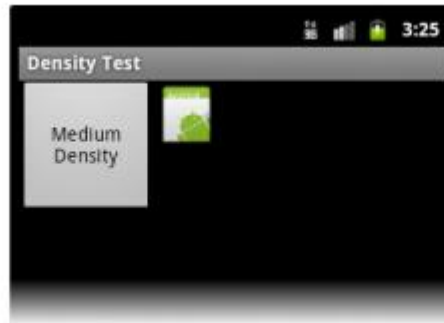
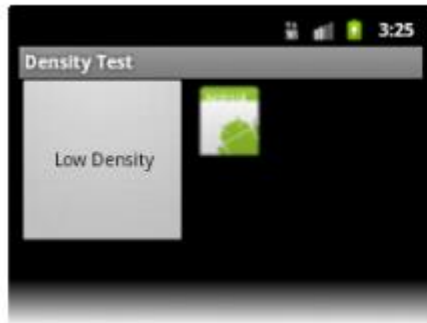
- A. Az Android automatikusan átméretezi a képet, ha nincs megfelelően illeszkedő.
- B. Az Android támogatja a sűrűségfüggetlen megjelenítést.
- C. $px = dp * (dpi / 160)$
- D. Közvetlenül pixelben nem adhatók meg a méretek.

Sűrűség függetlenség

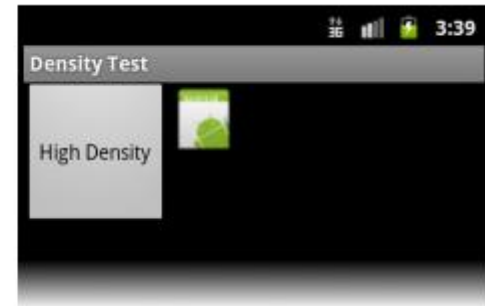
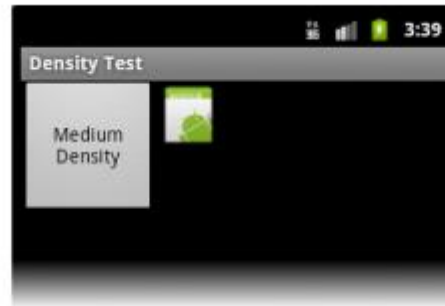
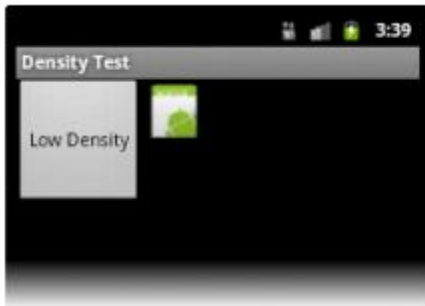
- Az alkalmazás akkor lehet „sűrűség független”, ha a felhasználói felületi elemek a felhasználó szemszögéből megőrzik a fizikai méretüket különböző sűrűségeken
- A „sűrűség függetlenség” fenntartása nagyon fontos, hiszen például egy gomb fizikailag nagyobbnak tűnhet egy alacsonyabb sűrűségű képernyőn
- A képernyő sűrűséghez kapcsolódó problémák jelentősen befolyásolhatják az alkalmazás felhasználhatóságát.
- Az Android kétféle módon is segít elérni a sűrűség függetlenséget:
 - > A rendszer a **dp** kiszámítása alapján skálazza a felhasználói felületet az aktuális képernyő sűrűségnek megfelelően
 - > A rendszer a képernyő sűrűség alapján automatikusan átskálazza a kép erőforrásokat

Példa

- Sűrűség függetlenség támogatás nélkül:



- Sűrűség függetlenség támogatással:



Kép erőforrások átméretezése

- Nem szerencsés, ha a rendszerre bízunk az átméretezést, hiszen így elmosódottak lehetnek a képek nagy felbontáson
- Az Android úgynevezett minősítő „string” (configuration qualifier)-ek segítségével teszi lehetővé, különböző erőforrások használatát
- A minősítő „string”-et az erőforrás könyvtár (res/) neve után kell fűzni (<resources_name>-<qualifier>, pl. *layout-xlarge*):
 - > <resources_name>: standard erőforrás típus, pl. *drawable*, vagy *layout*
 - > <qualifier>: minősítő a képernyőre vonatkozólag, pl. *hdpi*, vagy *large*
 - > Több minősítő is szerepelhet egymás után kötőjellel elválasztva

Legfontosabb minősítő értékek

- Méret:
 - > small, normal, large, xlarge
- Sűrűség:
 - > ldpi, mdpi, hdpi, xhdpi, nodpi (a rendszer az ebben lévőket nem méretezi át), tvdpi
- Irány:
 - > land, port
- Képarány:
 - > long (a jelentősen szélesebb, vagy magasabb kijelzőkhöz), notlong

Példák

- *res/layout/my_layout.xml*
 - *res/layout-small/my_layout.xml*
 - *res/layout-large/my_layout.xml*
 - *res/layout-xlarge/my_layout.xml*
 - *res/layout-xlarge-land/my_layout.xml*
-
- *res/drawable-mdpi/my_icon.png*
 - *res/drawable-hdpi/my_icon.png*
 - *res/drawable-xhdpi/my_icon.png*

Erőforrás választó algoritmus

- Futás közben egy meghatározott logika alapján választ a rendszer
- Megkeresi a passzoló erőforrást
- Ha nincs az aktuálshoz passzoló, akkor egy kisebb/alacsonyabb sűrűségűt választ (pl. *large* mérethez *normal* méretet választ)
- Amennyiben az elérhető erőforrások csak nagyobb képernyőkhöz vannak, mint a készülék képernyője, akkor hibát jelez az alkalmazás
- Például ha az összes egy típusú erőforrás *xlarge*-al van megjelölve, akkor *normal* képernyős eszközökön hiba keletkezik

Mire érdemes figyelni?

- Összefoglalva, az alkalmazásnak biztosítani kell, hogy:
 - > Elfér és használható kis képernyőkön
 - > Kihaszználja a nagy képernyőket és a rendelkezésre álló teret
 - > Álló és fekvő mód megfelelően kezelve van
- Bitmap-ok esetén érdemes a 3:4:6:8 skálázási arányt követni, pl:
 - > 36x36 low-density
 - > 48x48 medium-density
 - > 72x72 high-density
 - > 96x96 extra high-density

Tipikus méretek

- **320dp**: tipikus telefon képernyő (240x320 ldpi, 320x480 mdpi, 480x800 hdpi, stb).
- **480dp**: kisebb tabletek(480x800 mdpi).
- **600dp**: 7" tablet (600x1024 mdpi).
- **720dp**: 10" tablet (720x1280 mdpi, 800x1280 mdpi, stb.).

Például:

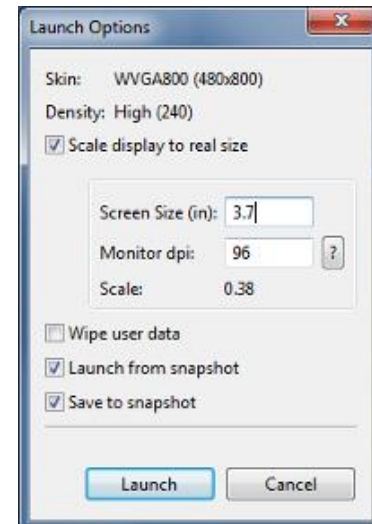
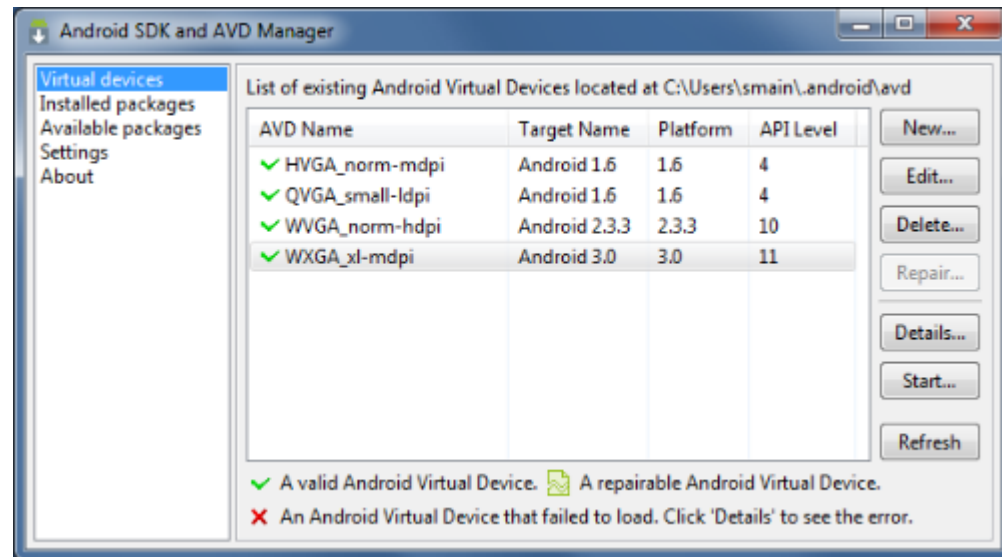
- > res/layout/main_activity.xml # Mobilok számára (kevesebb mint 600dp szélességgel)
- > res/layout-sw600dp/main_activity.xml # 7" tabletek számára (600dp széles, vagy nagyobb)
- > res/layout-sw720dp/main_activity.xml # 10" tabletek számára (720dp széles, vagy nagyobb)

Mi igaz az Android UI támogatására?

- A. Az Android nem támogatja a sűrűségfüggetleneséget.
- B. Ha nincs a képernyő tulajdonságaihoz illeszkedő erőforrás direkt megadva, akkor kivétel dobódik.
- C. A dp mértékegység helyett a dpi-t javasolt használni.
- D. Az Android futás közben tudja kikeresni a leginkább illeszkedő erőforrást.

Felhasználói felület tesztelése

- Az alkalmazás kiadása előtt mindenképp tesztelni kell a felhasználói felületet
- Az Android SDK támogat különféle emulátor skin-eket
- Tetszőlegesen beállítható az emulátor mérete, felbontása és pixelsűrűsége
- A teszteléshez létre kell hozni több AVD-t
- Skálázás és sűrűség megadása parancssorból indítás esetén (scale: 0.1-3):
 - > emulator -avd <avd_name>
-scale 96dpi



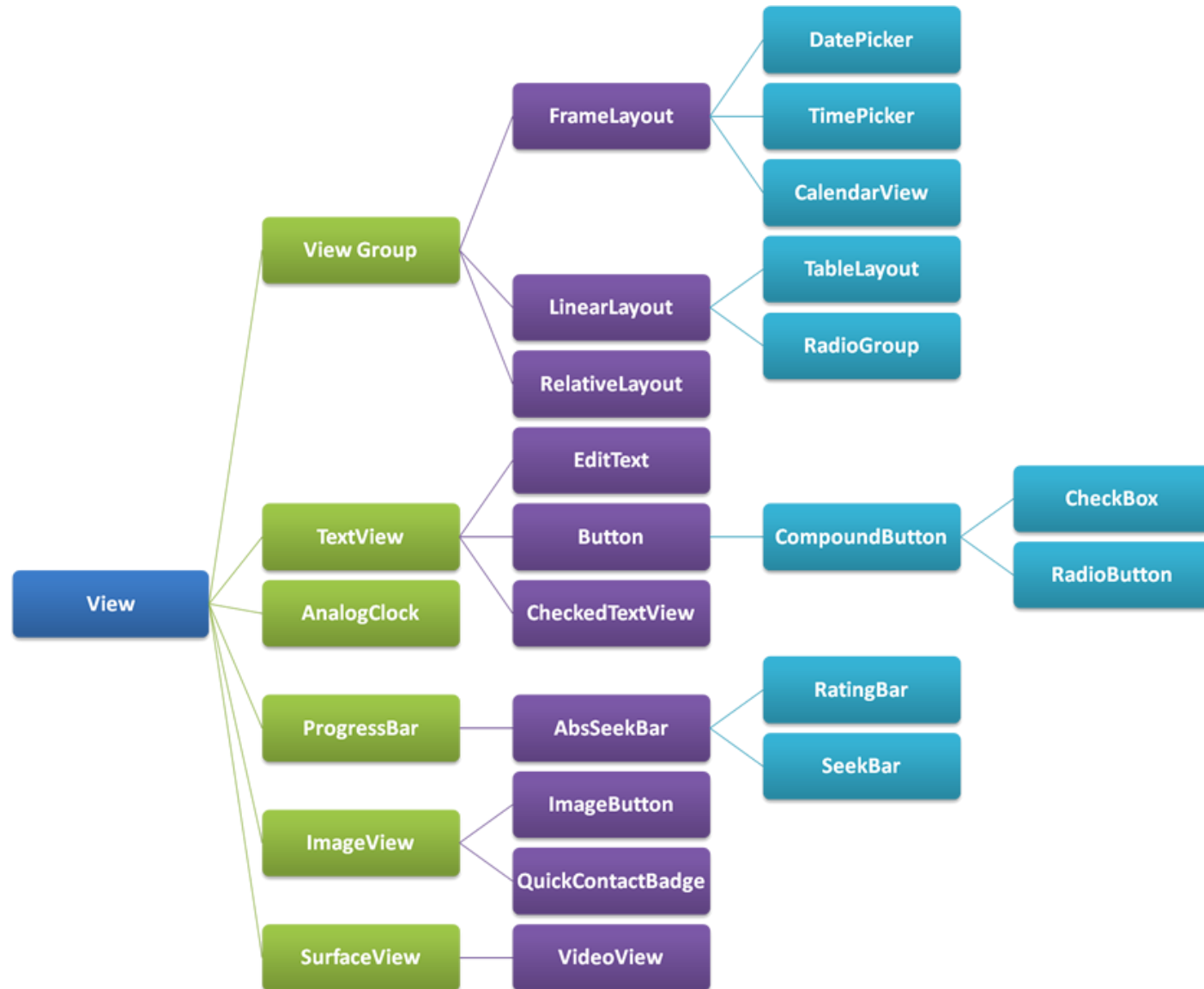
Felhasználói felület tervezése

- Elrendezés és erőforrások definiálása XML-ben
- UI erőforrás hozzárendelése Activity-hez
- Felületei elemek elérése forráskódból
 - > *findViewById([erőforrás azonosító – R.id.X])*
- Dinamikus felhasználói felület kezelés
- Animációk támogatása

Felhasználói felület erőforrások

- Felületek
 - > *res/layout*
- Szöveges erőforrás:
 - > *res/values/strings.xml*
- Kép erőforrások:
 - > *res/drawable-xyz/[kep].[ext]*
- Animáció erőforrások
 - > *res/anim*
- További erőforrások: *animator, szín, menü, nyers (raw), xml fileok*

Android UI architektúra



Layout-ok

Android felhasználói felület felépítése

- Minden elem a View-ból származik le
- Layout-ok (elrendezések):
 - > ViewGroup leszármazottak
 - > ViewGroup is a View-ból származik le!
- ViewGroup-ok egymásba ágyazhatók
- Saját View és ViewGroup is készíthető, illetve a meglevők is kiterjeszthetők

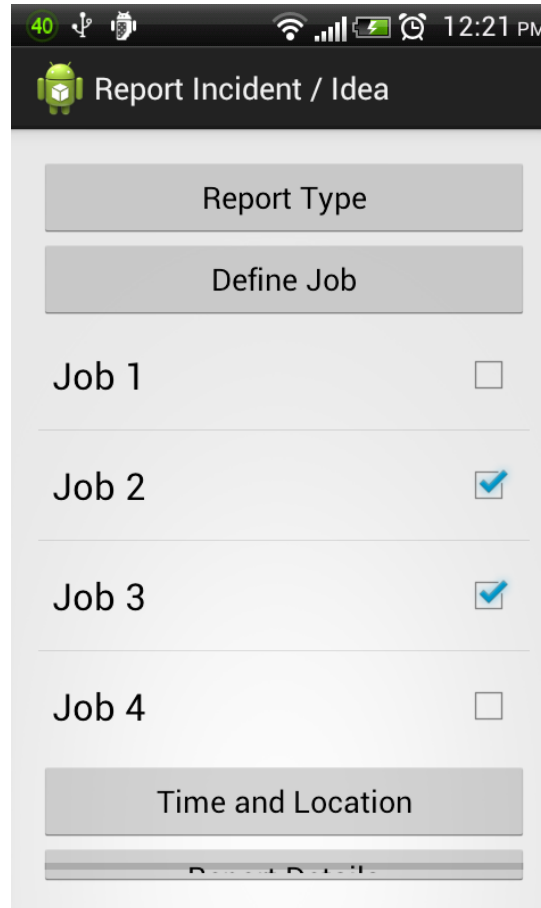
Layout-ok (ViewGroup)

- LinearLayout
- RelativeLayout
- ConstraintLayout
- AbsoluteLayout (NEM használjuk!)
- GridLayout
- RecyclerView
- Teljes lista:
 - > <http://developer.android.com/reference/android/view/ViewGroup.html>

```
public class  
RelativeLayout  
extends ViewGroup  
  
java.lang.Object  
└─ android.view.View  
    └─ android.view.ViewGroup  
        └─ android.widget.RelativeLayout
```

LinearLayout

- LinearLayout != Lista



The screenshot shows an Android application interface with a status bar at the top displaying 40% battery, signal strength, and the time 12:21 PM. The app title bar reads 'Report Incident / Idea' with an Android icon. The main content area is a vertical list of four items, each with a text label and a checkbox on the right. The items are 'Job 1', 'Job 2', 'Job 3', and 'Job 4'. 'Job 2' and 'Job 3' have their checkboxes checked with blue checkmarks. Below the list is a button labeled 'Time and Location'. At the very bottom, there is a partially visible button labeled 'Cancel'.

Job	Selected
Job 1	<input type="checkbox"/>
Job 2	<input checked="" type="checkbox"/>
Job 3	<input checked="" type="checkbox"/>
Job 4	<input type="checkbox"/>

Súlyozás Layout tervezéskor

- Megadható egy layout teljes súly értéke (weightSum)
- Elemek súly értéke megadható és az alapján töltődik ki a layout
 - > layout_weight érték
 - > A megfelelő width/height ilyenkor 0dp legyen!
- Hasonló, mint HTML-ben a %-os méret megadás

Layout súlyozás példa

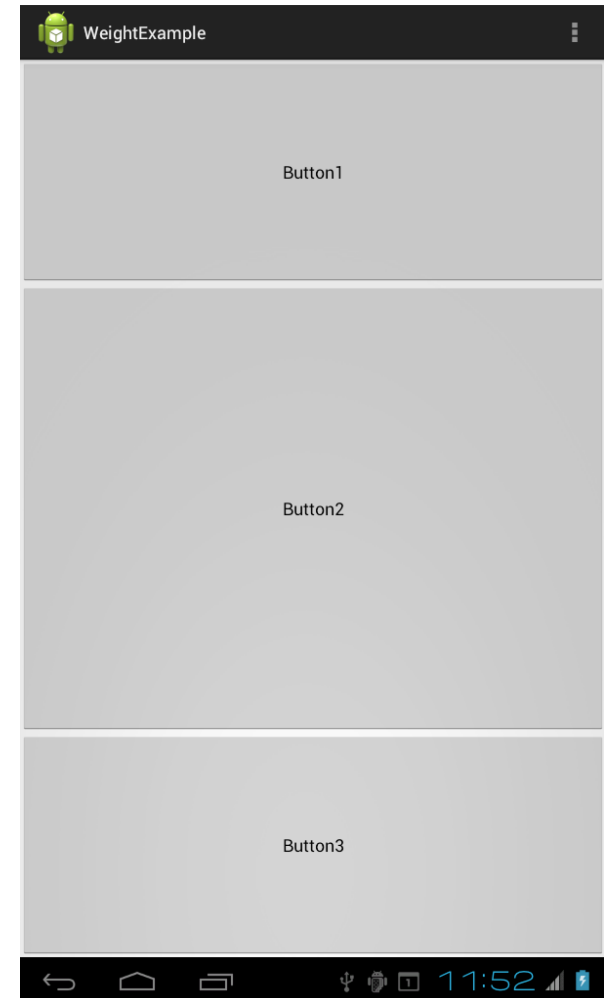
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:text="Button1" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:text="Button2" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:text="Button3" />
```

```
</LinearLayout>
```

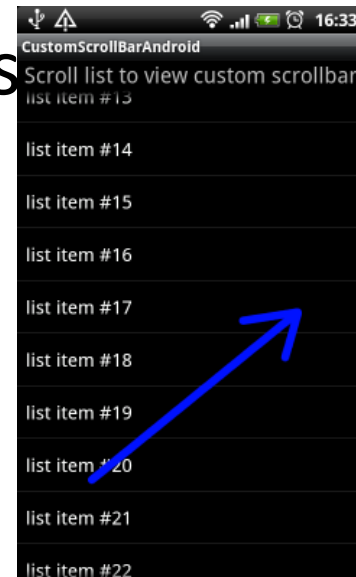


LinearLayout példák

- Jellemző paraméterek:
 - > Margin, padding
 - > Gravity
 - > ScrollView
 - > Weight

ScrollView

- ScrollView és HorizontalScrollView
- Layout container, amely scrollozást tesz lehetővé, ha a benne levő tartalom „nagyobb”
- Nem kötelező a teljes képernyőt kitöltenie
- Egy layout/képernyő több ScrollView-t is tartalmazhat



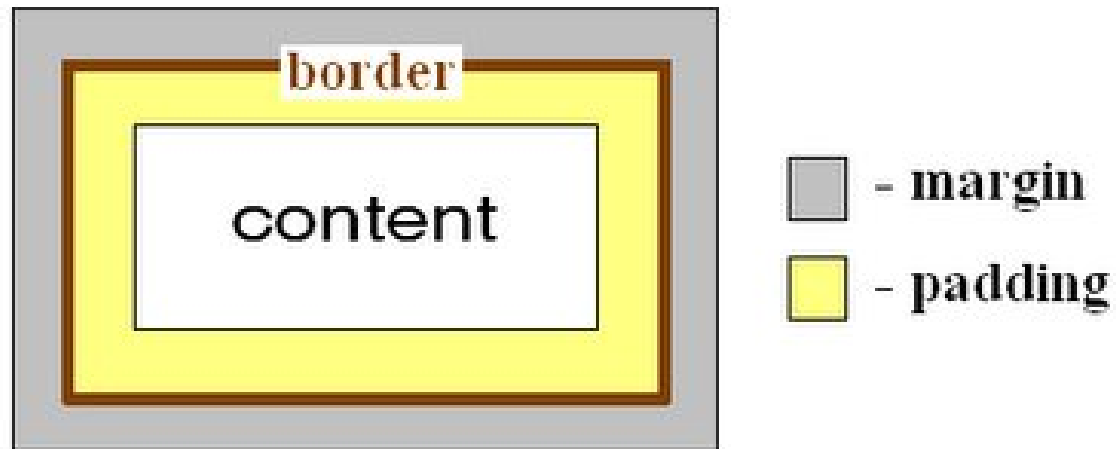
ScrollView példa

```
<ScrollView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            android:src="@drawable/image" />

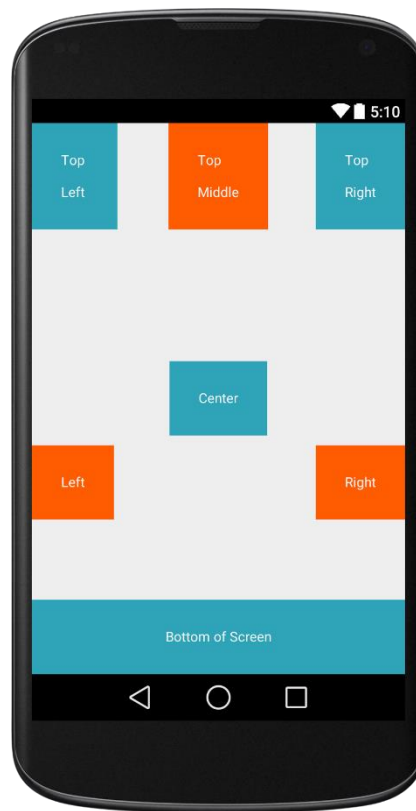
            ...
        </LinearLayout>
    </ScrollView>
```

Padding és Margin



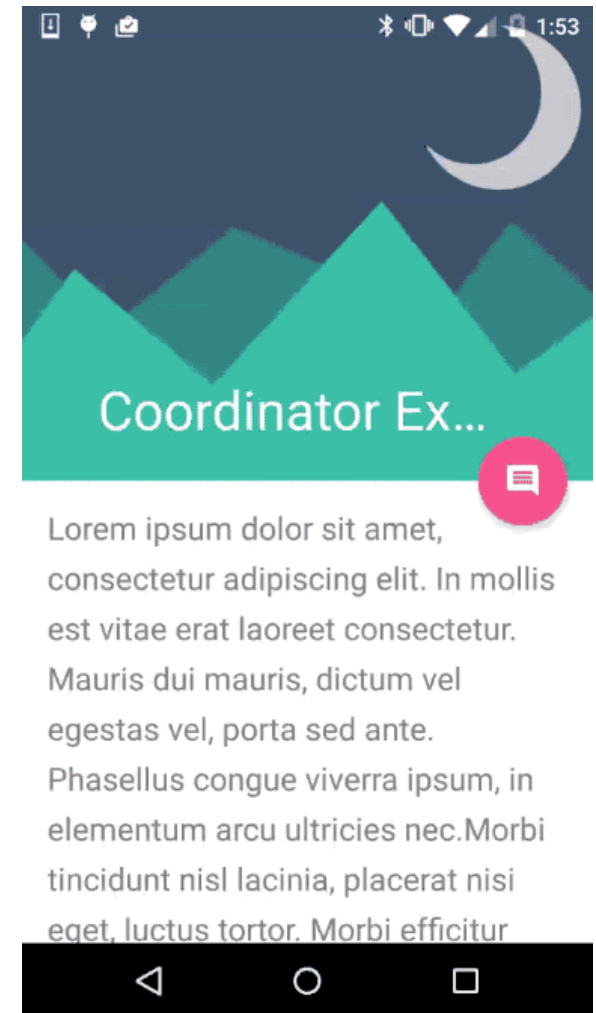
RelativeLayout

- Elemek egymáshoz való viszonya definiálható
- Demo



CoordinatorLayout, AppBarLayout

- CoordinatorLayout: továbbfejlesztett FrameLayout
- CoordinatorLayout fő feladatai:
 - > Felső szintű alkalmazás UI irányelv
 - > Konténer, mely támogatja a beépített elemek material stílushoz igazodó elhelyezkedését
- Behavior paraméterekkel meghatározható a kapcsolódó elemek elhelyezése
- AppBarLayout csatolható hozzá, mely a material design-hez illeszkedő scrollozást támogatja



CONSTRAINTLAYOUT

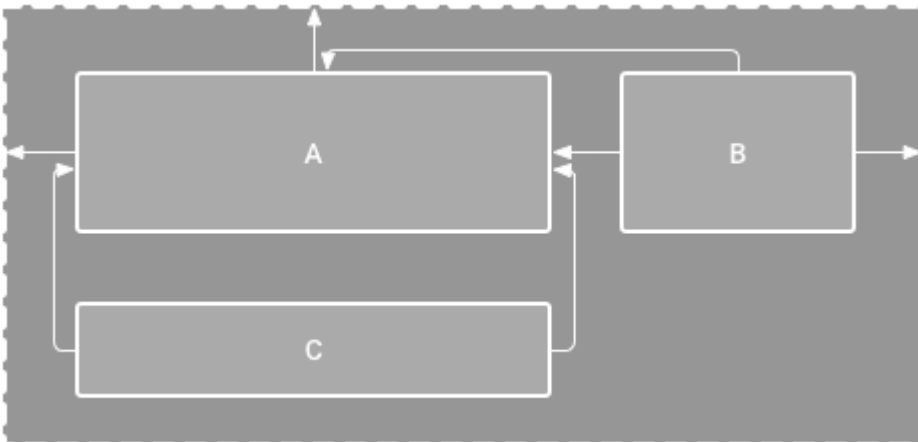
Reszponzív felületek ConstraintLayout-al

- Összetett, komplex layout-ok flat view hierarchiával
 - > 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>

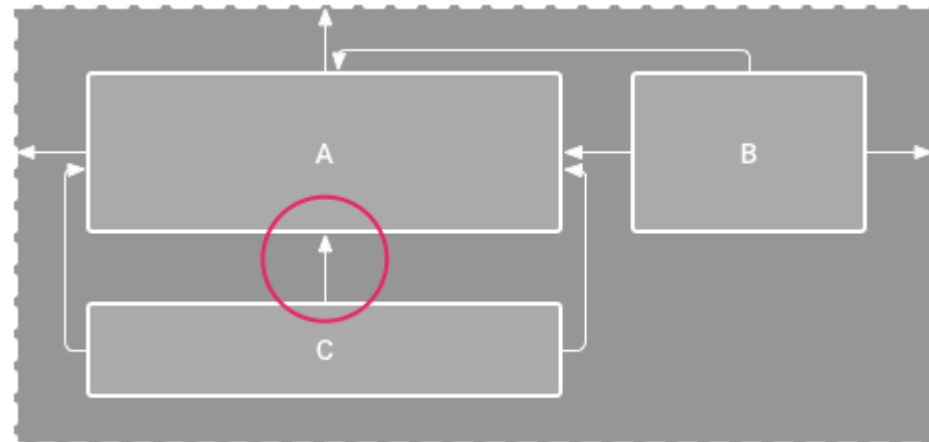
Áttekintés

- 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:

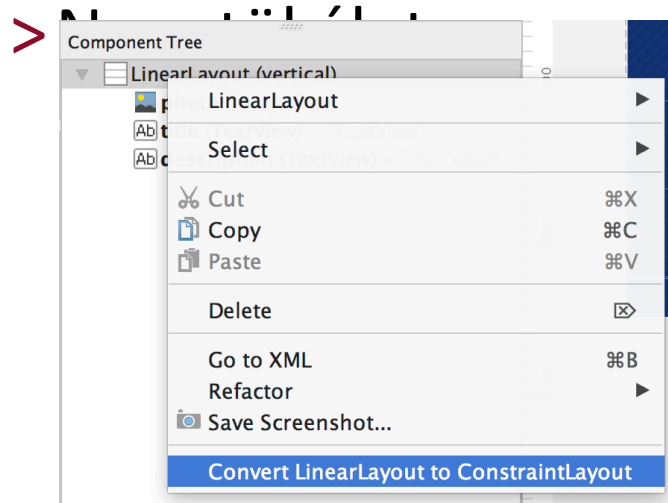


Helyes, mert C tudja, hogy A alatt van:



ConstraintLayout eszközök

- Gradle import:
 - > compile
'com.android.support.constraint:constraint-layout:1.0.2'
- Automatikus átalakítás

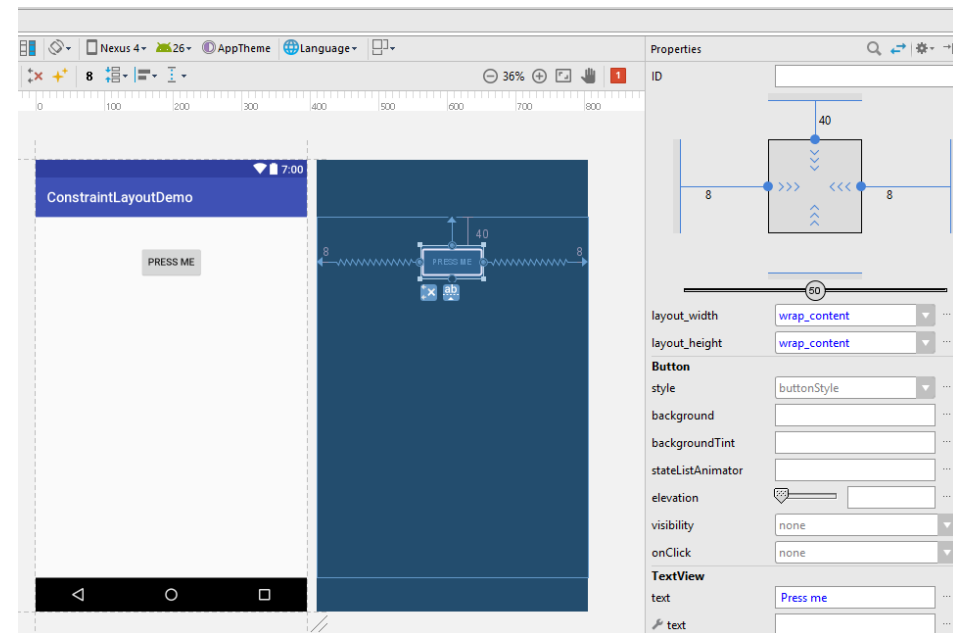


ConstraintLayout használat


- 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>
```

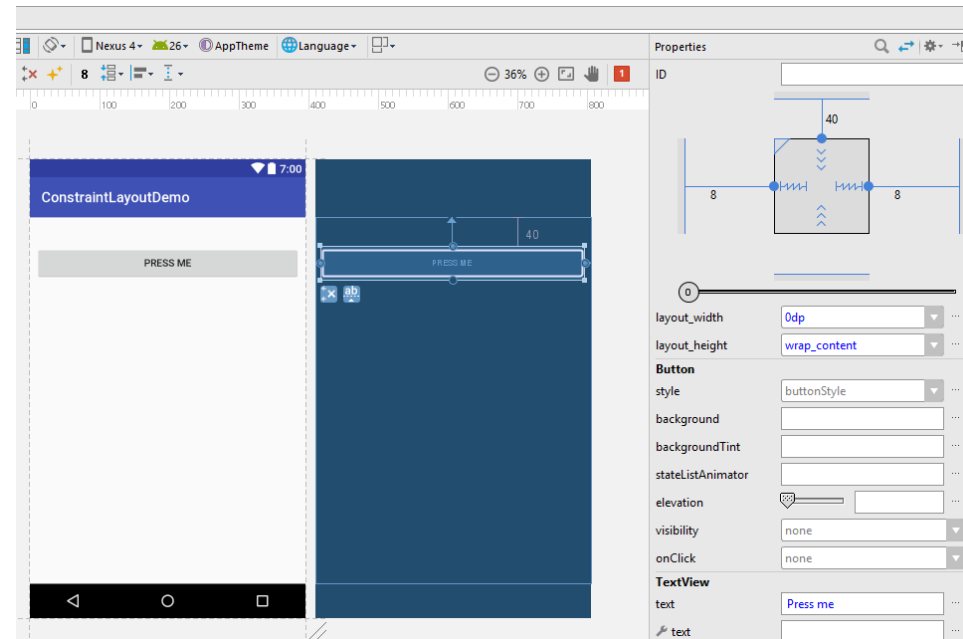


Mekkora lesz a gomb mérete?

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

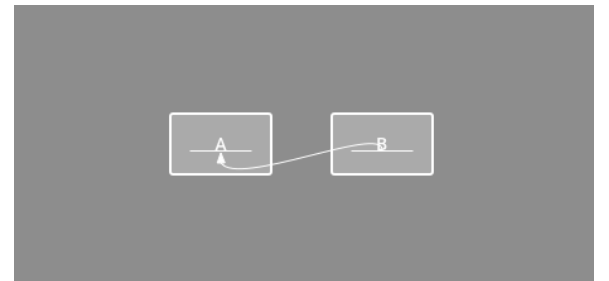
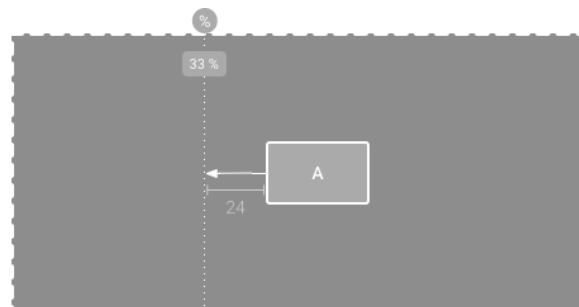
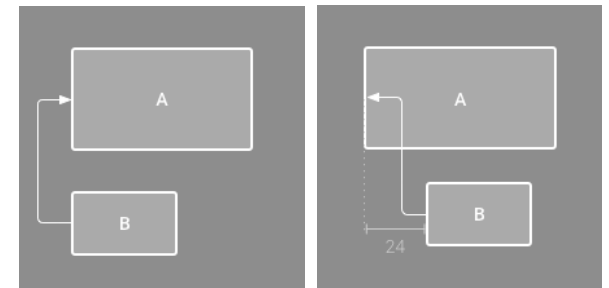
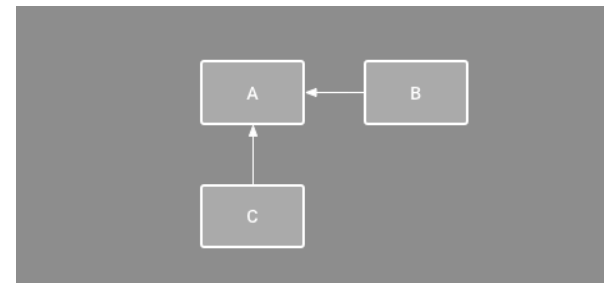
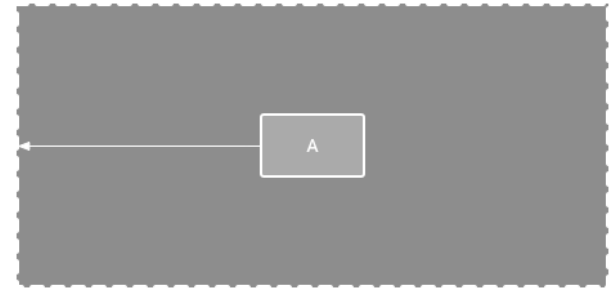
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
  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



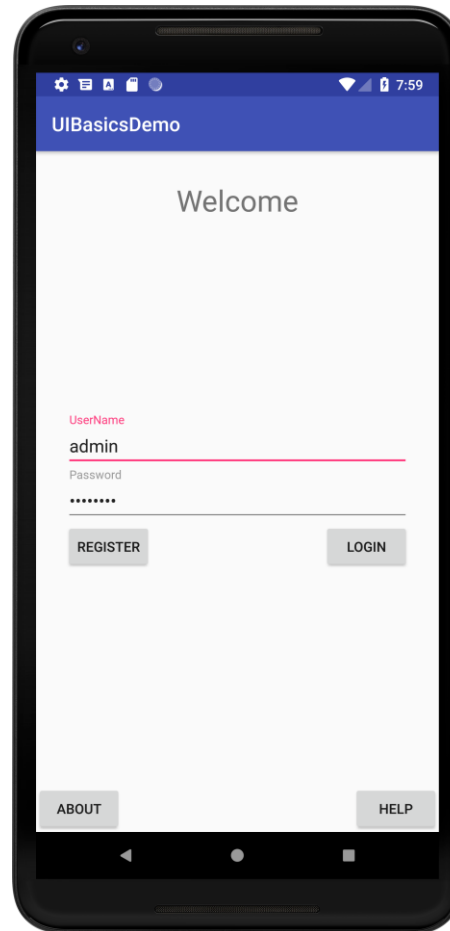
ConstraintLayout teljesítmény

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

Házi feladat



- Készítsünk egy Login képernyőt



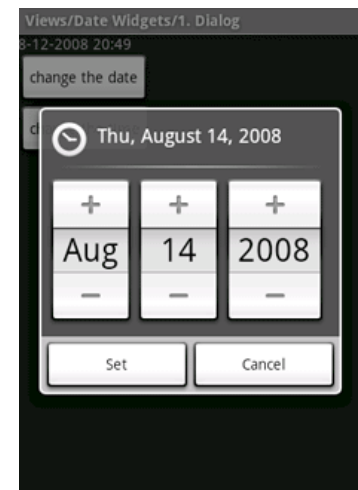
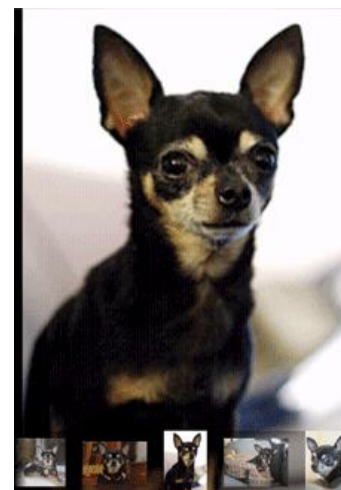
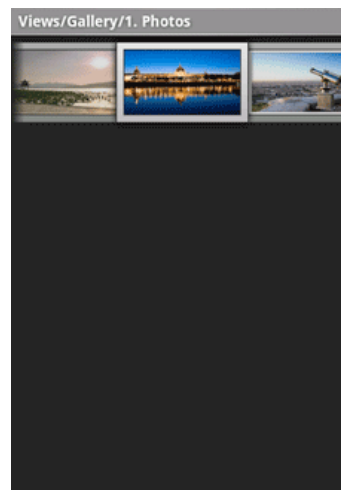
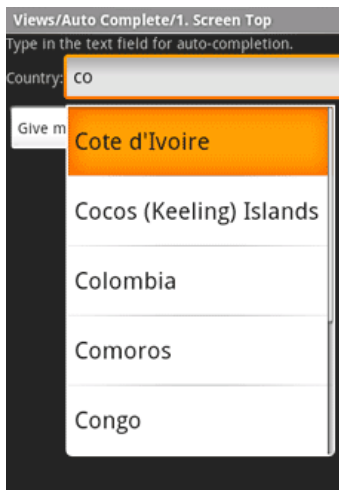
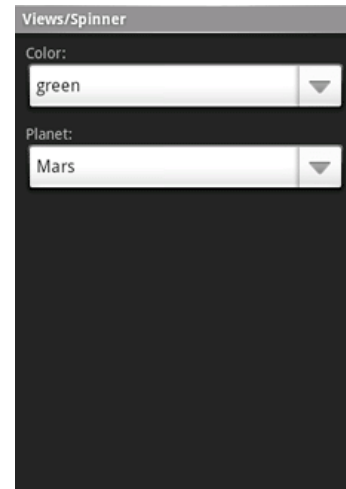
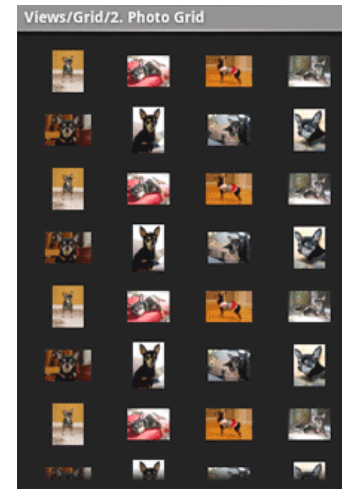
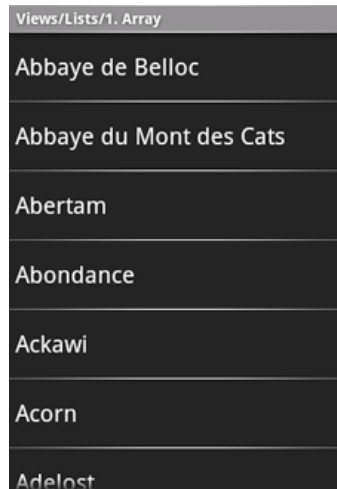
Nézetek (Widgetek/”View”-k)

View-k 1/2



- *Button, EditText, CheckBox, RadioButton, ToggleButton*
- ImageButton*
- ListView*
- GridView*
- Spinner*
- AutoCompleteTextView*
- Gallery*
- ImageSwitcher*
- DatePicker, TimePicker*

View-k 2/2



API gazdagsága (globálisan igaz az Androidra)

- Hogy valósítanak ezt meg? (nem sok *TextView* egymás után😊)



Lorem ipsum dolor sit amet

- Megoldás:
 - > <http://developer.android.com/reference/android/text/SpannableString.html>
 - > <http://androidcocktail.blogspot.hu/2014/03/android-spannablestring-example.html>

Egyedi nézetek – külső könyvtárak

- <https://github.com/wasabeef/awesome-android-ui/>

Összefoglalás

- Kotlin emlékeztető
- Alkalmazás komponensek
- Felhasználói felület tervezés és eszközök
- Layout-ok
 - > LinearLayout
 - > RelativeLayout
 - > ConstraintLayout
- Egyszerű View elemek

Köszönöm a figyelmet!



peter.ekler@aut.bme.hu