

Mobil alkalmazásfejlesztés - UI alapok - 1

Dr. Bilicki Vilmos

Szoftverfejlesztés Tanszék



A fóliához felhasznált anyagok:

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

Összefoglaló

- ▶ UI elrendezések és erőforrások
 - Nézet, Nézet csoport, nézet hierarchia
 - Elrendezés szerkesztő, ConstraintLayout
 - Eseménykezelés
 - Erőforrások és mérésük
- ▶ Szöveg és gördülő nézetek
- ▶ Activity-k és intent-ek

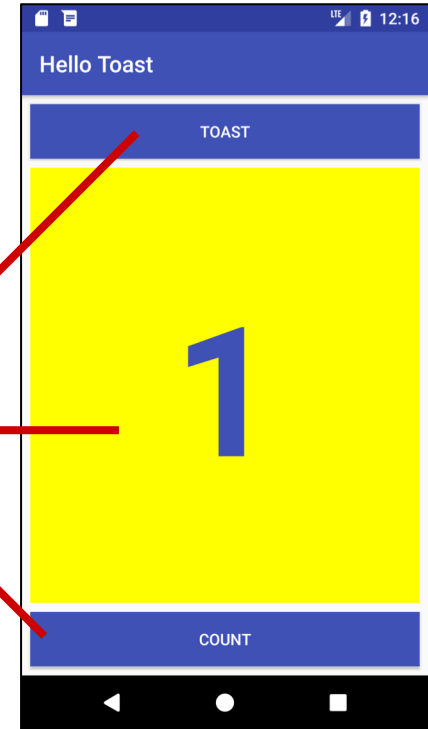




Amit látunk az nézet (view)

- Minden interfész elem egy-egy nézet

Nézetek



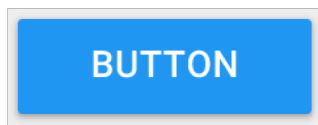
Mi a nézet?

- ▶ A felhasználói interfész alap blokkjai a View osztályai
 - Szöveg megjelenítés (TextView osztály), szöveg szerkesztése text (EditText osztály)
 - Gombok (Button class), menük, egyéb vezérlők
 - Görgethetők (ScrollView, RecyclerView)
 - Képek megjelenítése (ImageView)
 - Csoport nézetek (ConstraintLayout and LinearLayout)

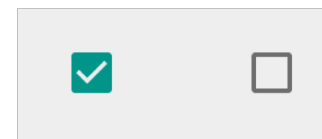


Minta nézet alosztályok

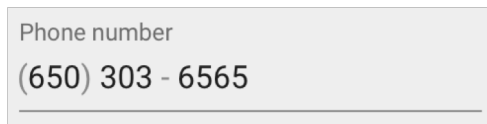
Gomb



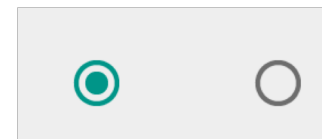
Jelölnégyezet



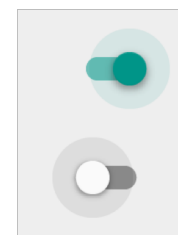
Szöveg
szerk.
Csúszka



Rádiógomb



Kapcsoló



Nézet attribútumok

- ▶ Szín, dimenzió, pozíció
- ▶ Lehet-e fókuszbán
- ▶ Lehet-e interaktív
- ▶ Látható-e
- ▶ Kapcsolata a többi nézettel

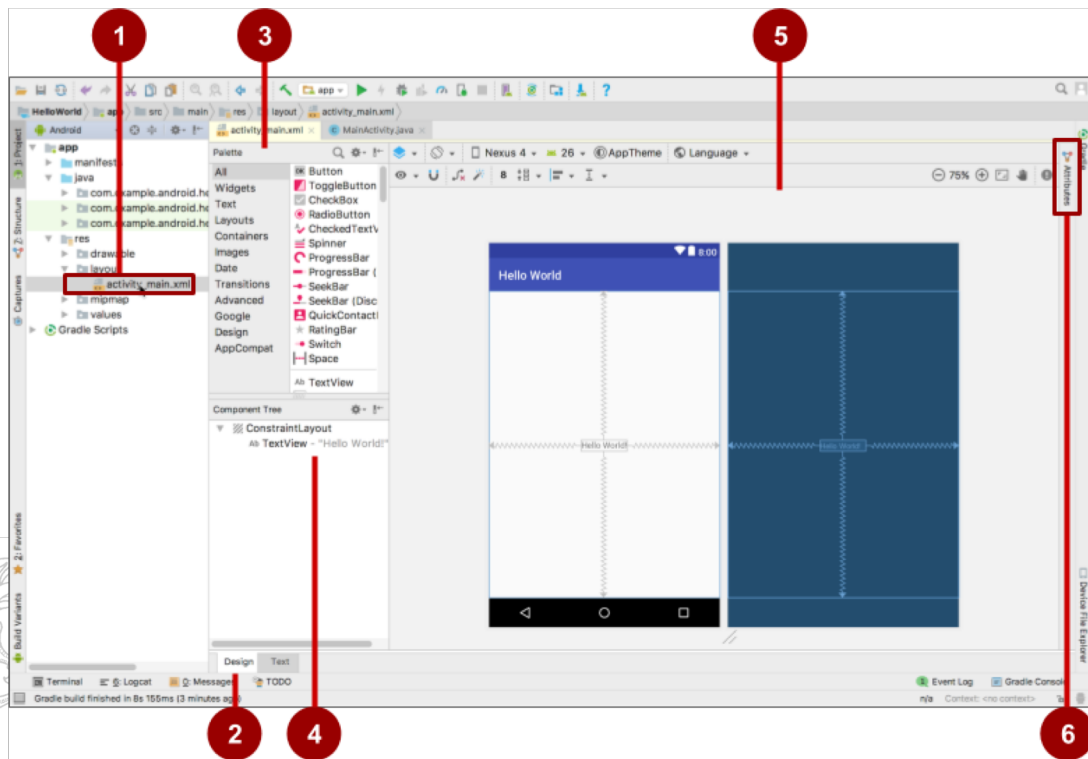


Nézetek és elrendezések létrehozása

- ▶ Android Studio elrendezés szerkesztő: XML vizuális reprezentációja
- ▶ XML szerkesztő
- ▶ Java kód



Android Studio elrendezés szerkesztő



1. XML elrendezés fájl
2. Tervezés és szöveg fülek
3. Paletta ablak
4. Komponens fa
5. Tulajdonságok fül

XML-ben definiált nézet

<TextView

android:id="@+id/show_count"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:background="@color/myBackgroundColor"

android:text="@string/count_initial_value"

android:textColor="@color/colorPrimary"

android:textSize="@dimen/count_text_size"

android:textStyle="bold"

/>



XML-ben specifikált attribútumok

android:<property_name>=<property_value>

Példa: android:layout_width="match_parent"

android:<property_name>="@<resource_type>/resource_id"

Példa: android:text="@string/button_label_next"

android:<property_name>="@+id/view_id"

Példa: android:id="@+id/show_count"



Java kódból

kontextus

In an Activity:

```
TextView myText = new TextView(this);  
myText.setText("Display this text!");
```



Mi a kontextus?

- ▶ Context egy interfész az alkalmazás környezetének leíró globális környezethez
- ▶ A kontextus elkérése:

```
Context context = getApplicationContext();
```

- ▶ Az Activity saját kontextusában:

```
TextView myText = new TextView(this);
```



Egyedi nézetek

- ▶ Több mint 100 különböző a View osztályból származtatott nézet
- ▶ Bármelyik alosztályaként létrehozható saját egyedi nézet



ViewGroup gyermek nézeteket tartalmaz

- ▶ ConstraintLayout: Az UI elemeket adott egymáshoz és az elrendezés széleihez viszonyított kényszerek mentén helyezi el
- ▶ ScrollView: egy elemet tartalmaz és megengedi a görgetést
- ▶ RecyclerView: egy elem listát tartalmaz és a listát tudja bővíteni csökkenteni és görgetni



ViewGroup az elrendezésekhez

- ▶ Layout – elrendezés
 - A ViewGroup alosztályai
 - Gyermek nézeteket tartalmaznak
 - Lehetnek: sor, oszlop, rács, tábla és abszolút elrendezésűek



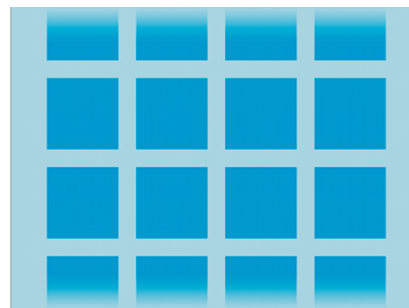
Gyakori elrendezés osztályok



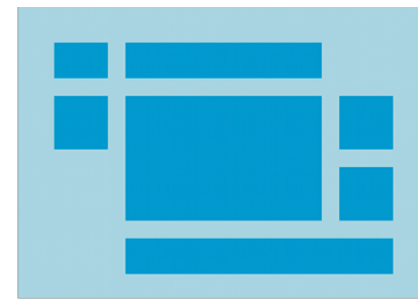
LinearLayout



ConstraintLayout



GridLayout

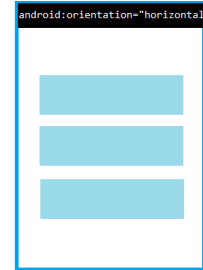
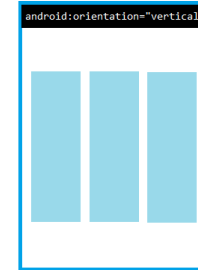
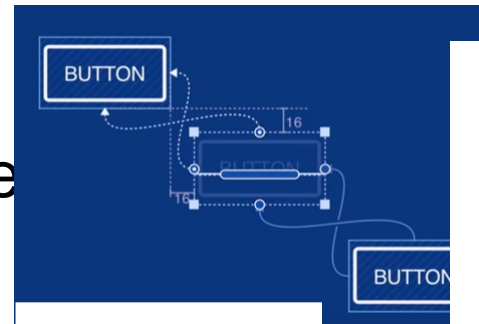


TableLayout



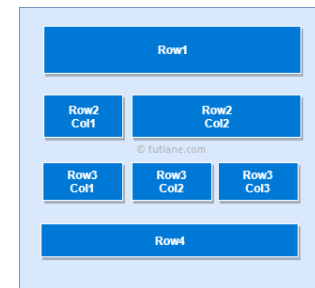
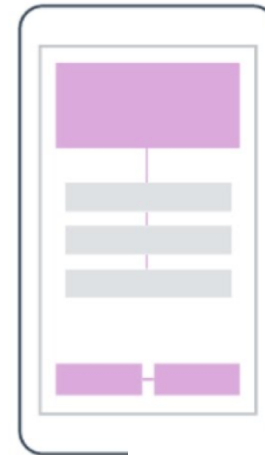
Gyakori elrendezés osztályok

- ▶ **ConstraintLayout:** kényszerek segítségével köti össze a nézeteket
- ▶ **LinearLayout:** Horizontális vagy vertikális sor
- ▶ **RelativeLayout:** a gyermek nézetek relatívak egymáshoz
- ▶ **TableLayout:** Sorok és oszlopok
- ▶ **FrameLayout:** Egy gyermek egymásra helyezett gyermekekből



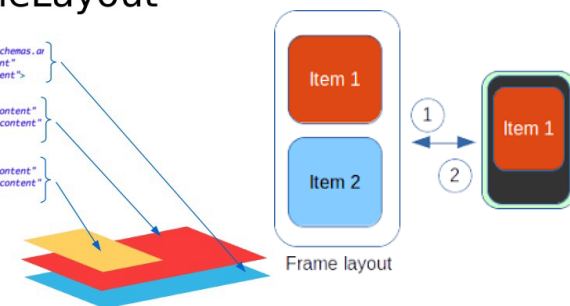
© www.androidinterview.com

www.facebook.com/androidinterview



FrameLayout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</FrameLayout>
```



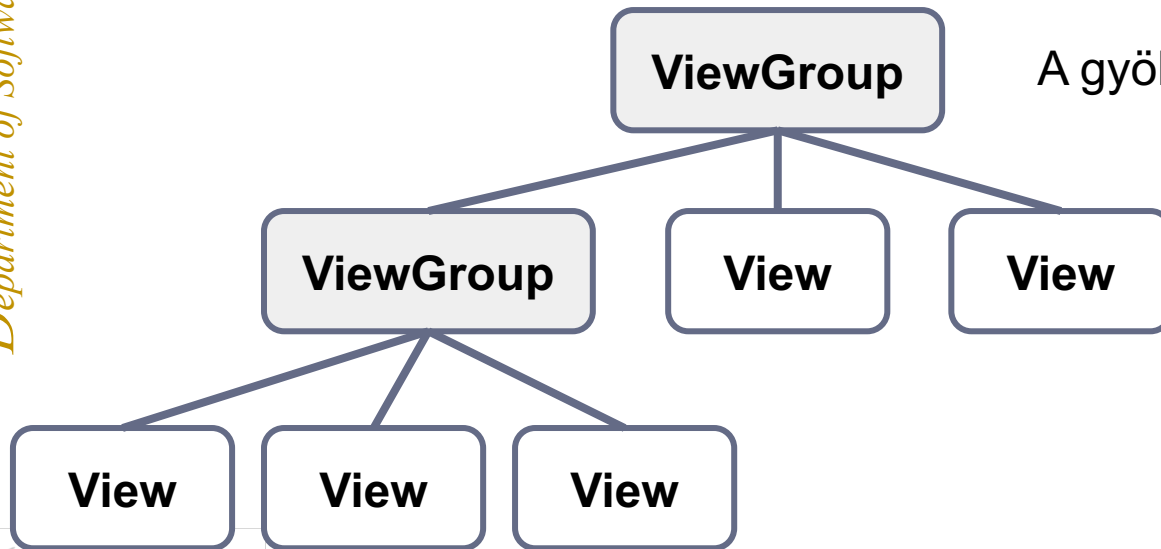
Frame layout

Osztály vs elrendezés hierrachia

- ▶ Nézet osztály hierarchia a klasszikus OO öröklődés
 - Pl.: Button és TextView amely View amely egy objektum
 - Szuperosztály-alosztály kapcsolat
- ▶ Nézet hierarchia a nézetek vizuális elrendezését mutatja meg
 - pl: LinearLayout sorbarendzett Button-okat tartalmazhat
 - Szülő – gyermek kapcsolat

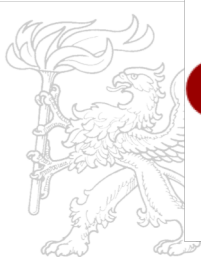


A nézet csoport és a nézetek hierarchiája

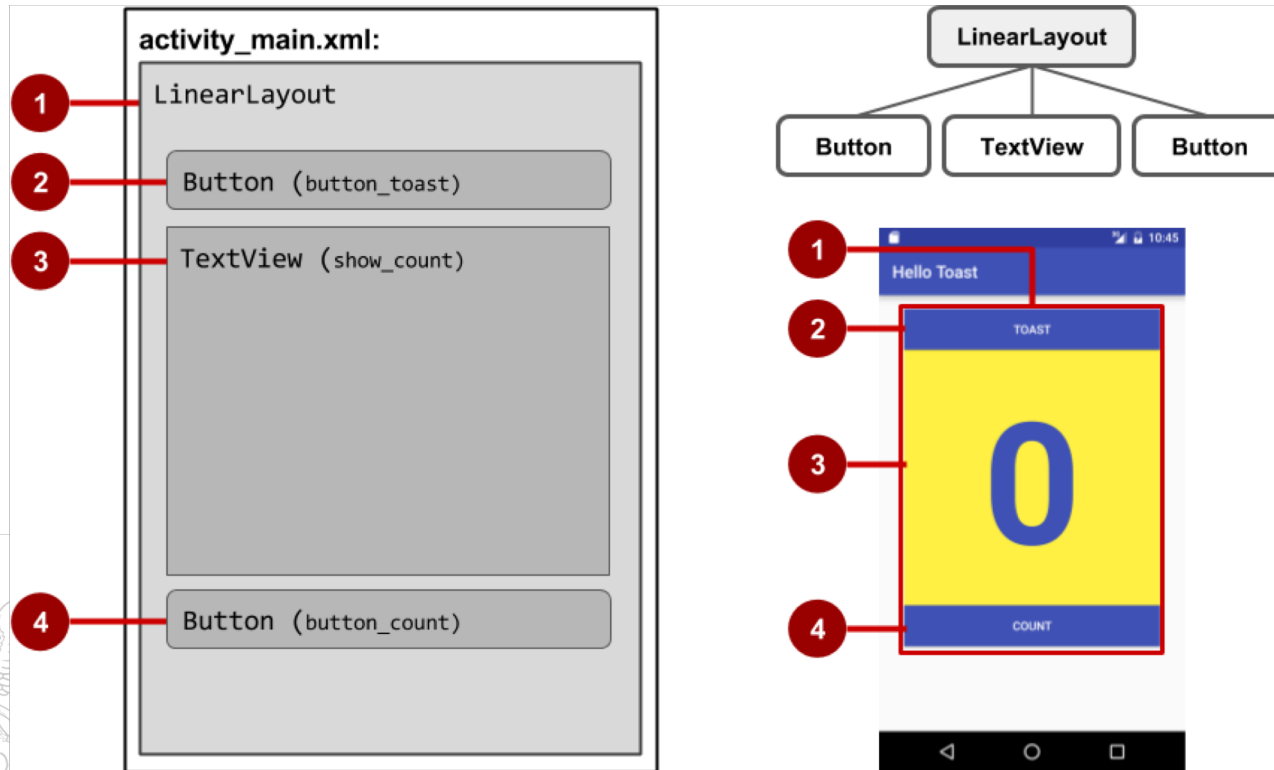


A gyökér nézet mindig ViewGroup



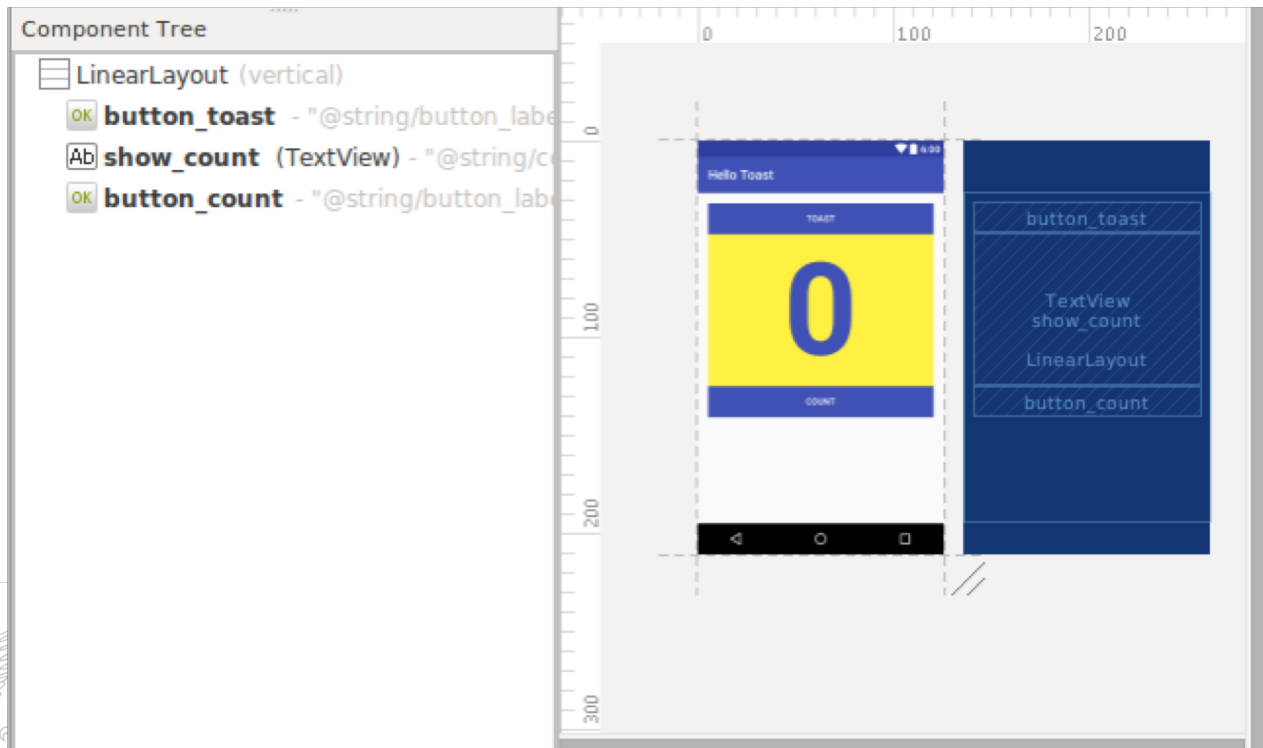


Nézet hierarchia és képernyő elrendezés





A nézet szerkesztőben



XML-ben

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        ... />
    <TextView
        ... />
    <Button
        ... />
</LinearLayout>
```



Java-ban

```
LinearLayout linearL = new LinearLayout(this);  
linearL.setOrientation(LinearLayout.VERTICAL);
```

```
TextView myText = new TextView(this);  
myText.setText("Display this text!");
```

```
linearL.addView(myText);  
setContentView(linearL);
```



Java-ban

Magasság és szélesség beállítása:

```
LinearLayout.LayoutParams
```

```
layoutParams =
```

```
    new LinearLayout.LayoutParams(
```

```
        LayoutParams.MATCH_PARENT,
```

```
        LayoutParams.MATCH_CONTENT);
```

```
myView.setLayoutParams(layoutParams);
```



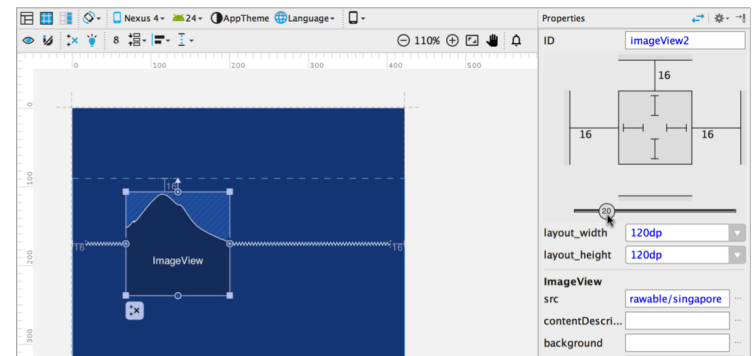
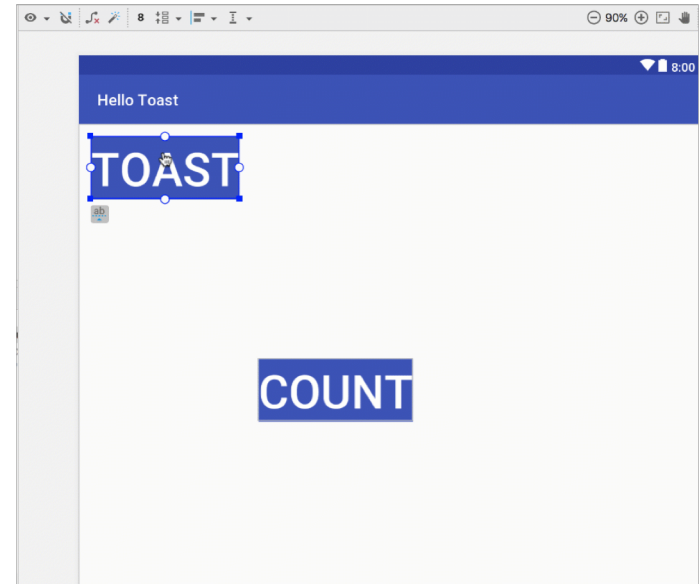
Legjobb gyakorlat

- ▶ A nézet hierarchia befolyásolja a teljesítményt
- ▶ A legkevesebb és legegyszerűbb nézeteket alkalmazzuk
- ▶ Laposra tervezzük – ne legyen mély a nézet – nézet csoport egymásba ágyazás



Elrendezés szerkesztő ConstraintLayout

- ▶ Az UI elemek szülőhöz kötése
- ▶ Elemek pozícionálása és átméretezése
- ▶ Elemek egymáshoz képesti elrendezése
- ▶ Dimenziók és szélek beállítása
- ▶ Tulajdonságok beállítása



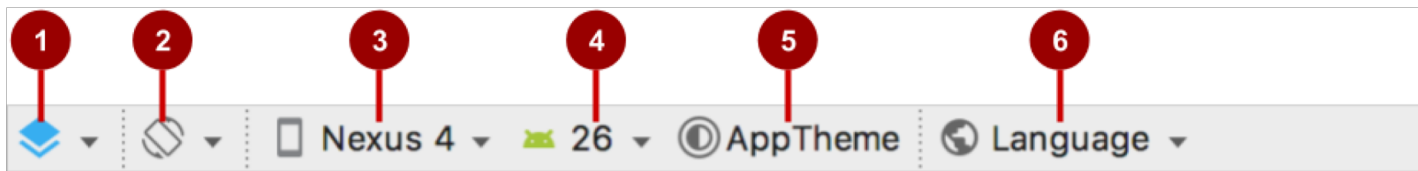
ConstraintLayout

- ▶ az Android Studio alapértelmezett elrendezése
- ▶ egy kellő flexibilitás nyújtó ViewGroup
- ▶ az elemek helyét és méretét kényszerek segítségével határozza meg
- ▶ a kényszer egy kapcsolat egy másik nézet, a szülő elrendezés vagy vezető elemhez

Android	iOS
top	top
bottom	bottom
left	left
right	right
start	leading
end	trailing
	centerX
	centerY
baseline	baseline



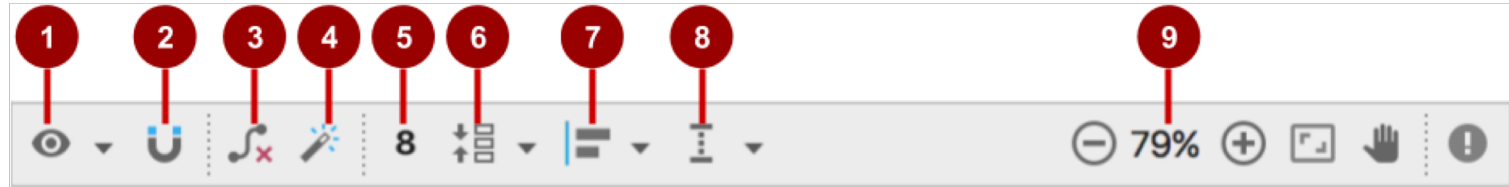
Elrendezés szerkesztő eszköztár



1. Tervező felület: Tervező és váz nézetek
2. Orientáció: Vízszintes és Függőleges
3. Eszköz: A megjelenítés biztosító eszköz
4. API verzió: Az előnézethez használt API
5. A szerkesztő témája: Az előnézet témája
6. Lokalizáció: Az előnézet lokalizációja, nyelve



Kényszer elrendezés szerkesztő eszköztár

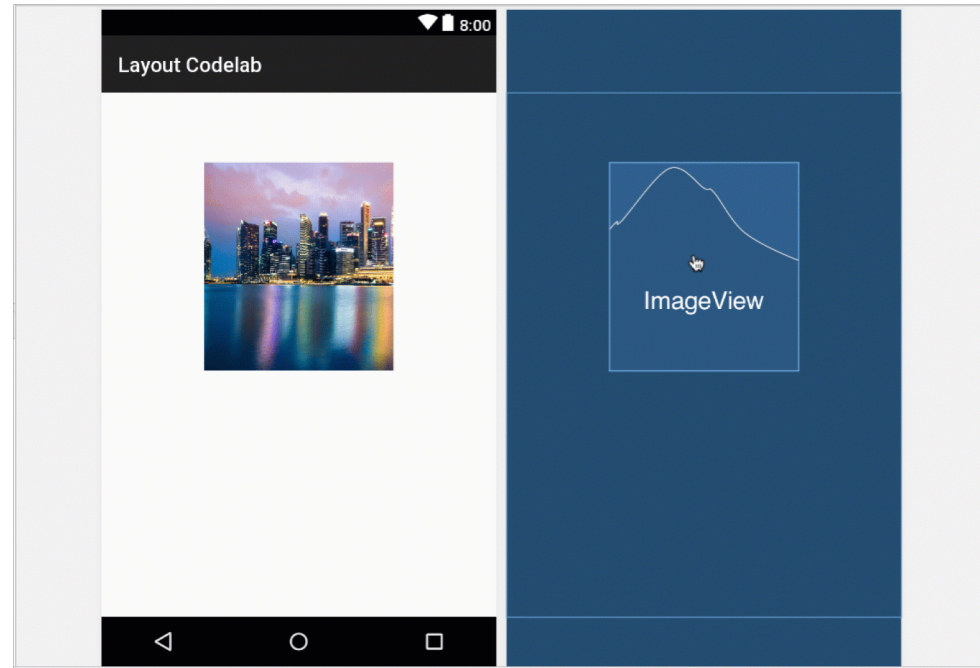


1. Megmutat: Megmutatja a kényszereket és a margókat
2. Automatikus huzalozó: Engedjük - tiltjuk
3. Minden kényszer törlése: Az elrendezés összes kényszerét törli
4. Kényszerek következtetése: Kényszerek létrehozása következtetéssel
5. Alapértelmezett margók: Beállítja az alapértelmezett margókat
6. Becsomagolás: Az adott elemeket becsomagolja vagy kibontja
7. Elrendezés: Elrendezi az adott elemeket
8. Vezérvonalak: Vertikális vagy horizontális vezérvonalak
9. Nagyítás vezérlők: Nagyítás szintje



Automatikus huzalozás

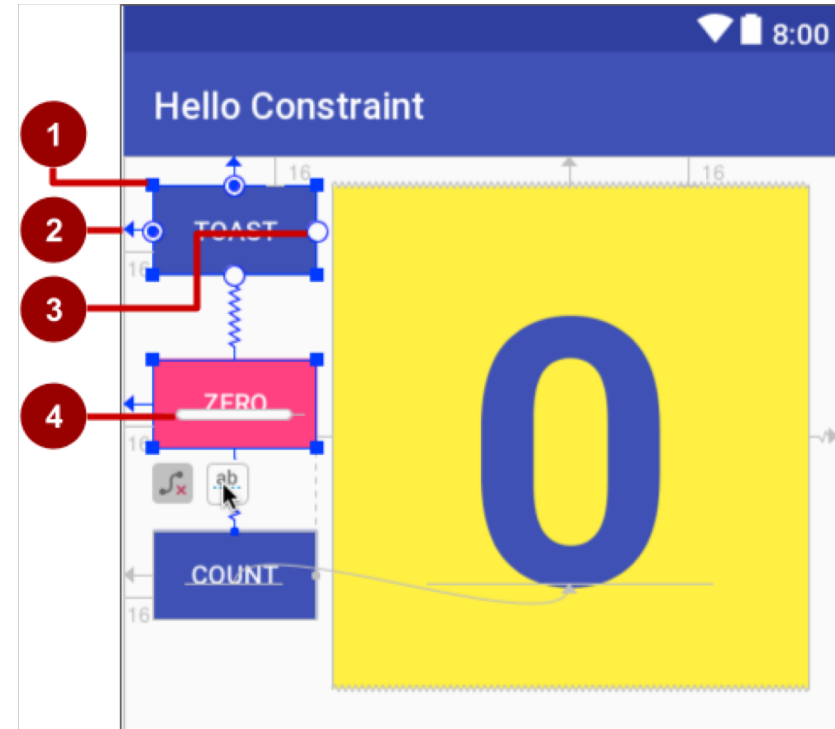
- ▶ Engedélyezzük az Autoconnect-et amennyiben titlva volt
- ▶ Húzzunk rá elemeket
- ▶ Az Autoconnect a szülőhöz viszonyítva generál kényszreket





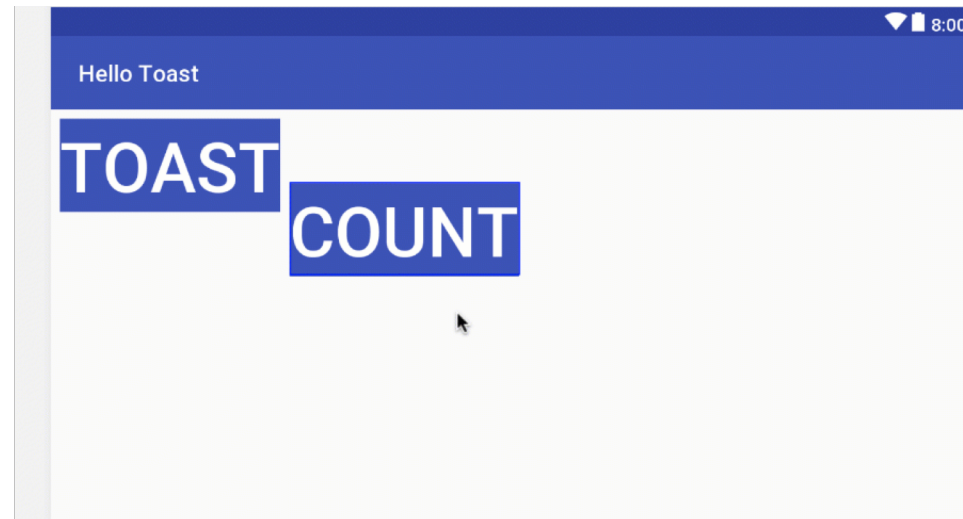
Kényszer elrendezés kezelők

1. Átméretezés
2. Kényszer vonal és kezelő
3. Kényszer kezelő
4. Alapvonal kezelő



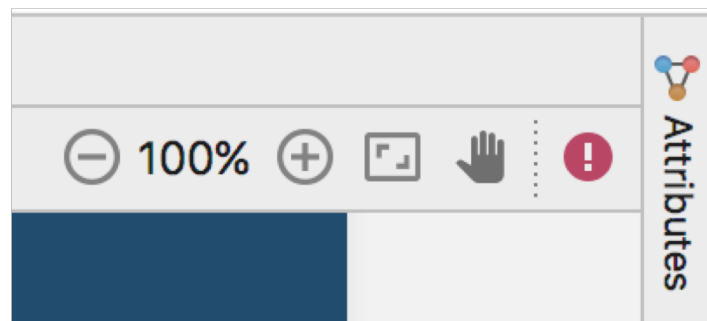
Alapvonalhoz rendezés

1. Kattintsunk az alapvonal kényszer gombra 
2. Húzzuk a másik elem alapvonalához



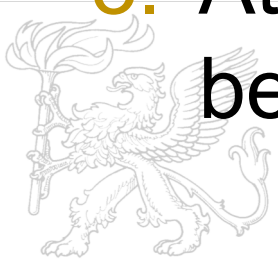
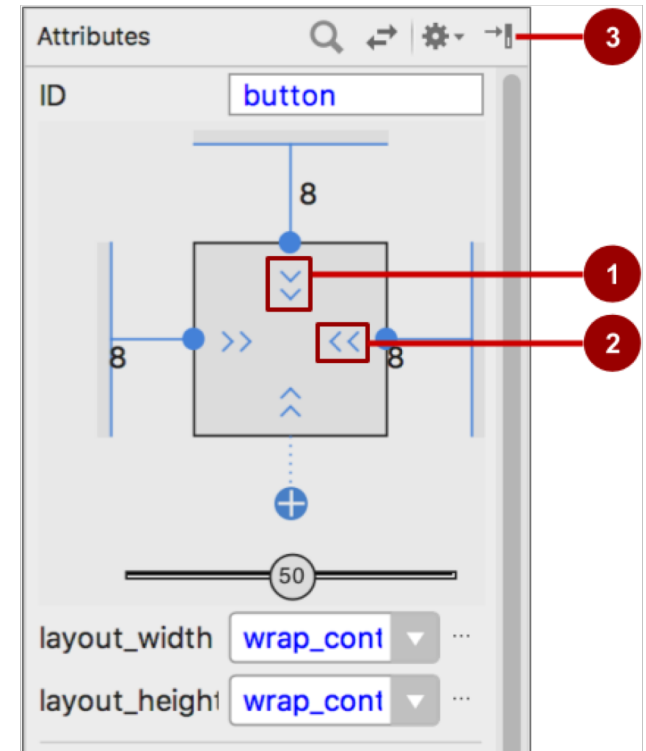
Tulajdonságok panel

- ▶ Margó vezérlők
- ▶ Olyan attribútumok mint `layout_width`



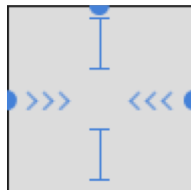
Attribútumok panel nézet kezelő

1. Vertikális nézet
méret
layout_height
2. Horizontális nézet
méretlayout_width
3. Attribútumok panle
becsukás gomb



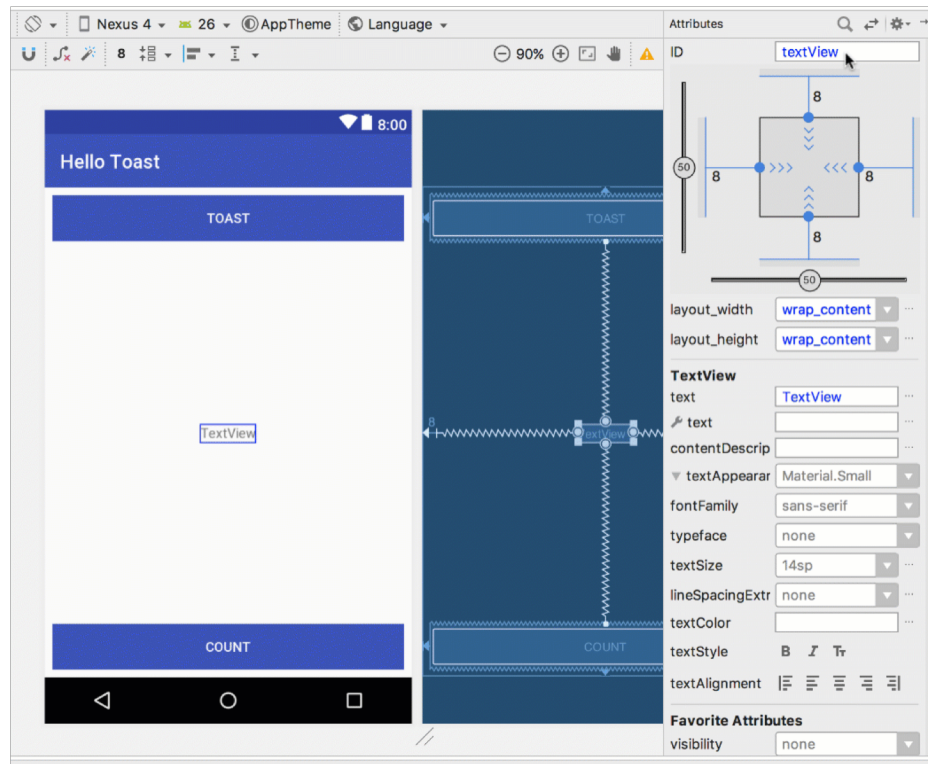
Layout_width és layout_height

- ▶ a layout_width és layout_height változik a méret vezérlőkkel
- ▶  match_constraint: Kiterjeszti a vezérlőt, hogy kitöltse a szülőt
- ▶  wrap_content: Leszűkíti a vezérlőt a befoglalt tartalomhoz
- ▶  Fix dp (density-independent pixels)

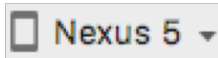




Atribútumok beállítása



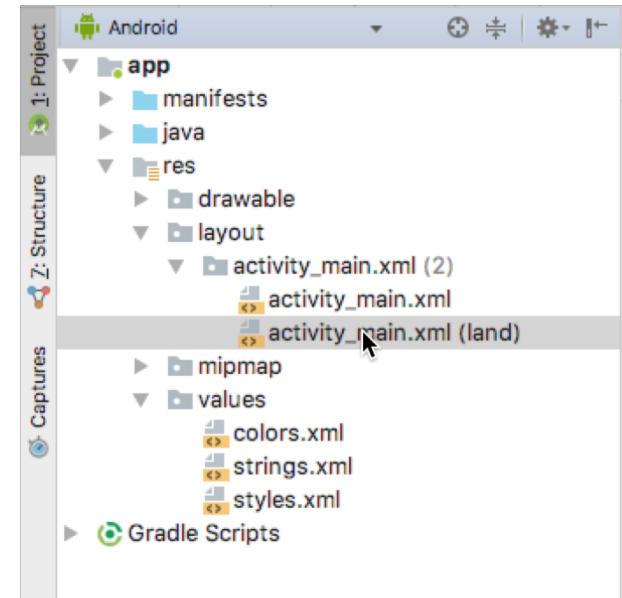
Előnézet megjelenítések

- ▶ Megnézhetjük horizontális és vertikális elrendezésben
 - Orientation gomb 
 - Válasszuk ki a megjelenítést
- ▶ Megnézhetjük különböző eszközön
 - Eszköz gomb 



Orientáció függő megjelenítés

1. Orientáció gomb
2. Create Landscape Variation
3. Létrehozza a activity_main.xml (land) megjelenítés variánst
4. Szerkesszük



Eszközfüggő megjelenítés

1. Orientáció gomb
2. Create layout x-large Variation
3. Létrehozza a activity_main.xml (xlarge) megjelenítés variánst
4. Szerkesszük



Események

▶ Valami történik

- A felületen: kattintás, húzás, érintés
- Az eszközzel: DetectedActivity: járás, autó vezetés, ...
- Az Android rendszer által detektált események

▶ Eseménykezelők

- Egy metódus melyet az esemény hatására meghívunk



XML-ben csatolt és Java-ban megvalósított

**Csatoljuk a
nézethez XML-
ben:**

android:onClick="showToast"



**Java activity-ben
megvalósítva:**

```
public void showToast(View view) {  
    String msg = "Hello Toast!";  
    Toast toast = Toast.makeText(  
        this, msg, duration);  
    toast.show();  
}
```

Java-ban

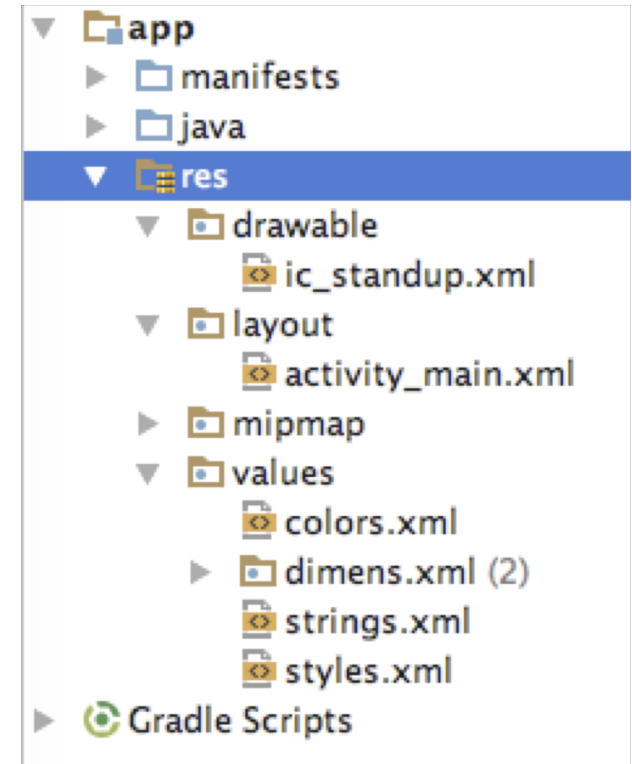
```
final Button button = (Button)
findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg,
duration);
        toast.show();
    }
});
```





Erőforrások

- ▶ Statikus adat és kód elválasztása
- ▶ Karatreláncok, képek, ...
- ▶ Lokalizációnál hasznos



Hivatkozás a kódból

- Elrendezés:
`R.layout.activity_main`
`setContentView(R.layout.activity_main);`
- Nézet:
`R.id.recyclerview`
`rv = (RecyclerView)`
`findViewById(R.id.recyclerview);`
- Karakterlánc:
Java: `R.string.title`
XML: `android:text="@string/title"`





Méretetek

► Jó

- Sűrűség független képpontok (dp): nézeteknek
- Skála független képpontok (sp) a szövegnek

► Rossz

- Aktuális képpontok (px)
- Aktuális mérték (mm, in)
- Pont alapú tipográfia 1/72 inch (pt)



Szöveg és görgetés nézetek

- ▶ A TextView egy View alosztály
 - egy és többsoros szöveget kezel
- ▶ Az EditText a TextView alosztálya
 - szerkeszthető szöveget kezel
- ▶ A megjelenítés attribútumokkal vezérlhetőek
- ▶ Szöveg megadása:
 - Statikusan az XML-ben lévő karakterlánc erőforrásból
 - Dinamikusan Java-ból



Szöveg formázása

- ▶ `<b>` és `<i>` elemeket használhatjuk
- ▶ karakterlánc erőforrások: egy bekezdés – nem tagolt
- ▶ `\n`-nel kezdődik
- ▶ Nem ASCII kezelése (`\`)
- ▶ Idézőjelek kezelése (`\'` , `\"`)



XML-ben

```
<TextView android:id="@+id/textview"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:text="@string/my_story"/>
```



Gyakori tulajdonságok

[android:text](#)—a szöveg

[android:textColor](#)—színe

[android:textAppearance](#)—stílusa

[android:textSize](#)—mérete sp-ben

[android:textStyle](#)—normal, bold, italic, or
bold|italic

[android:typeface](#)—normal, sans, serif, or
monospace

[android:lineSpacingExtra](#)—extra helyek a
sorok között



URL-ek formázása

```
<string name="article_text">... www.rockument.com ...</string>
```

```
<TextView  
    android:id="@+id/article"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:autoLink="web"  
    android:text="@string/article_text"/>
```

Ne használjunk
HTML-t a web
hivatkozásokhoz



Java

```
TextView myTextview = new TextView(this);  
myTextView.setWidth(LayoutParams.MATCH_PARENT);  
myTextView.setHeight(LayoutParams.WRAP_CONTENT);  
myTextView.setMinLines(3);  
myTextView.setText(R.string.my_story);  
myTextView.append(userComment);
```



Gördülő nézet

- ▶ Nagymennyiségű szöveg
- ▶ A TextView gögretéséhez bele kell tennünk egy ScrollView-be
- ▶ Csak egy gyerek eleme lehet
- ▶ Több gyermekhez használjunk ViewGroup-ot (pl.: LinearLayout)
- ▶ FrameLayout alosztálya
- ▶ A teljes tartalmat a memóriában tárlja
 - Nem alkalmas komplex tartalmak, nagy szövegek kezelésére
- ▶ Ne ágyazzunk gördülő nézeteket egymásba



Példa: egy TextView

```
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subhead
ing">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        .../>
</ScrollView>
```



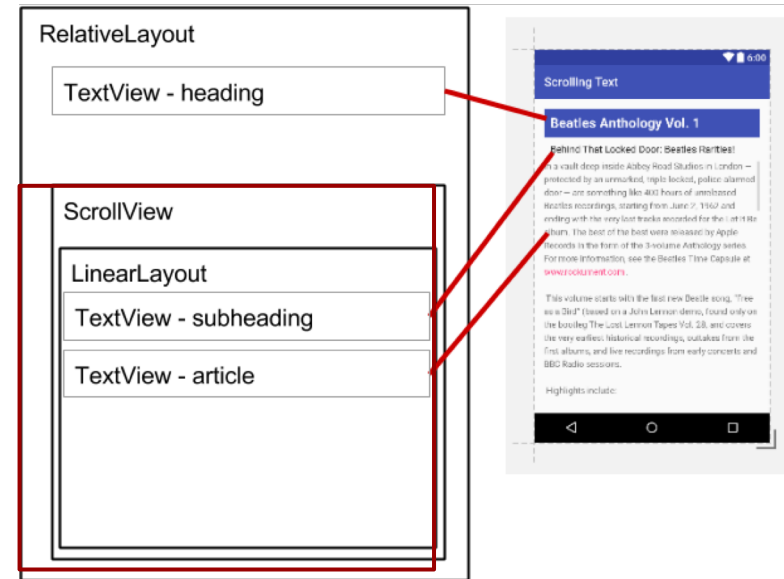
Példa: nézet csoport

```
<ScrollView ...
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
      android:id="@+id/article_subheading"
      .../>

    <TextView
      android:id="@+id/article" ... />

  </LinearLayout>
</ScrollView>
```



Képpel, gombbal

```
<ScrollView...>
```

```
    <LinearLayout...>
```

```
        <ImageView.../>
```

```
        <Button.../>
```

```
        <TextView.../>
```

```
    </LinearLayout>
```

```
</ScrollView>
```



A ScrollView egy gyermeke

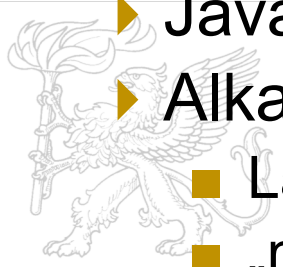


Az elrendezés gyermeke

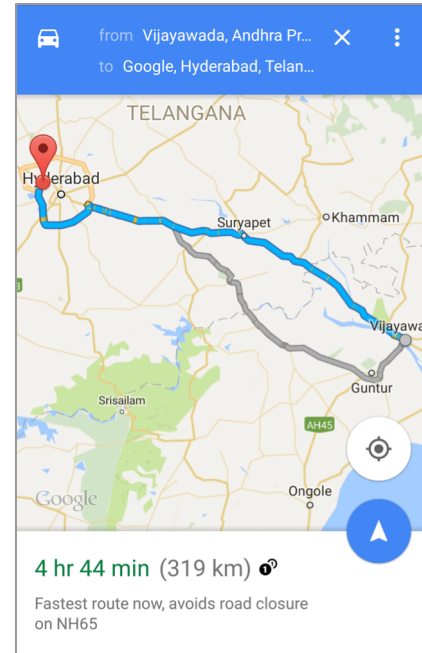
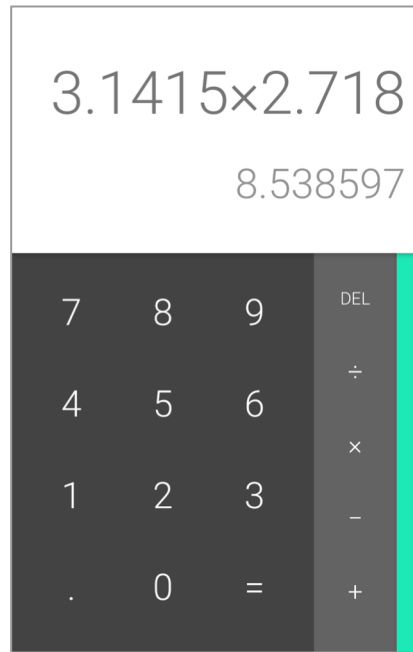
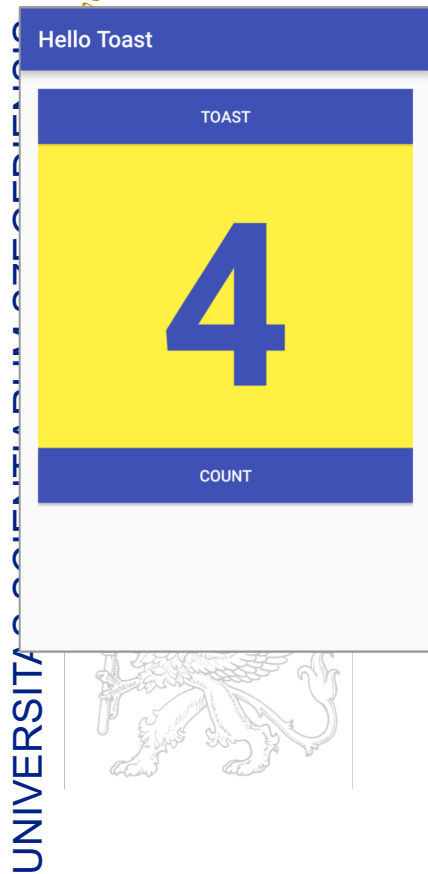


Aktivitások

- ▶ Az Activity egy alkalmazás komponens
 - Eseményeket kezel
 - Egy ablakot, egy nézet hierarchiát ábrázol
 - Indíthat más aktivitásokat
 - Életciklusa van
 - Általában van elrendezése (Layout)
- ▶ Általában kitölti a képernyőt, de lehet beágyazva, vagy lebeghet is
- ▶ Java osztály – tipikusan egy Activity egy fájl
- ▶ Alkalmazás vs. Activity
 - Lazán csatolva alkotnak egy alkalmazást
 - „main activity”
 - Hierarchiába szervezhetőek (navigációt segítő)



Példa



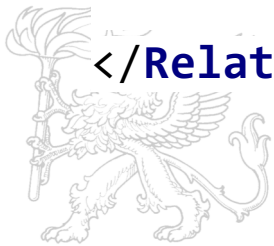
Activity megvalósítása

1. XML-ben kell definiálni
2. Activity Java osztály
 - AppCompatActivity-t egészíti ki
3. Kössük össze az Activity-t az elrendezéssel
 - onCreate() – content view
4. Deklaráljuk az Activity-t az Android leíróban



1. XML - elrendezés

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Let's Shop for Food!" />
</RelativeLayout>
```



2. Java osztály

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```



3. Kössük össze az elrendezéssel

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);
```

setContentView(R.layout.activity_main
);

↑ ↑ ↑
Erőforrás elrendezés ebben az XML-ben



4. Android leíró

```
<activity  
  android:name=".MainActivity">
```



4. Adjuk meg a fő aktivitás a leíró fájlban

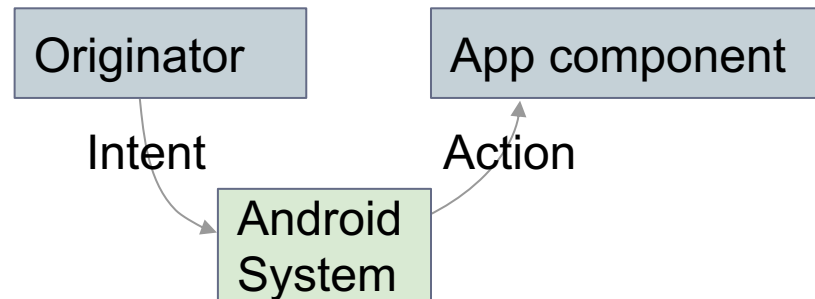
MainActivity-nek intent-filter-t is be kell illeszteni ahhoz, hogy elindítható legyen

```
<activity android:name=".MainActivity">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```



Intent

- ▶ Az Intent a végrehajtandó művelet leírása
- ▶ Az Intent segítségével kér az egyik alkalmazás komponens a másik alkalmazás komponensről műveletet az Android futtatási környezet segítségével



Inent-ek segítségével

- ▶ Indíthatunk Activity-ket
 - A gomb nyomás segítségével új Activity indul a szöveg bevitelre
 - A megosztás gombra az alkalmazás a megosztási lehetőségeket mutatja
- ▶ Elindíthatunk egy szolgáltatást
 - Háttérben letöltés elidítése
- ▶ Kézbesíthetünk üzenetszórást (Broadcast)
 - A rendszer értesít mindenkit, hogy a telefon most csörög



Explicit és implicit intent

► Explicit intent

- Egy meghatározott Activity-t indít el
 - Kávét kérünk Zsuzsától
 - A fő Activity a ViewShoppingCart Activity-t indítja el

► Implicit intent

- Megkérjük a rendszert, hogy keressen egy megfelelő Activity-t amely a kérését teljesítheti
 - Keressünk egy boltot ahol zöld teát vehetünk
 - A megosztás gombra kattintva a megosztást kezelő alkalmazások listáját látjuk



■ Activity indítása intent-tel

Explicit:

1. Intent-t létrehozása

- `Intent intent = new Intent(this, ActivityName.class);`

2. Indítsuk el vele az Activity-t

- `startActivity(intent);`

Implicit:

1. Intent-t létrehozása

- `Intent intent = new Intent(action, uri);`

2. Indítsuk el vele az Activity-t

- `startActivity(intent);`



Implicit Intent példák

Weboldal megnyitása

```
Uri uri = Uri.parse("http://www.google.com");  
Intent it = new Intent(Intent.ACTION_VIEW, uri);  
startActivity(it);
```

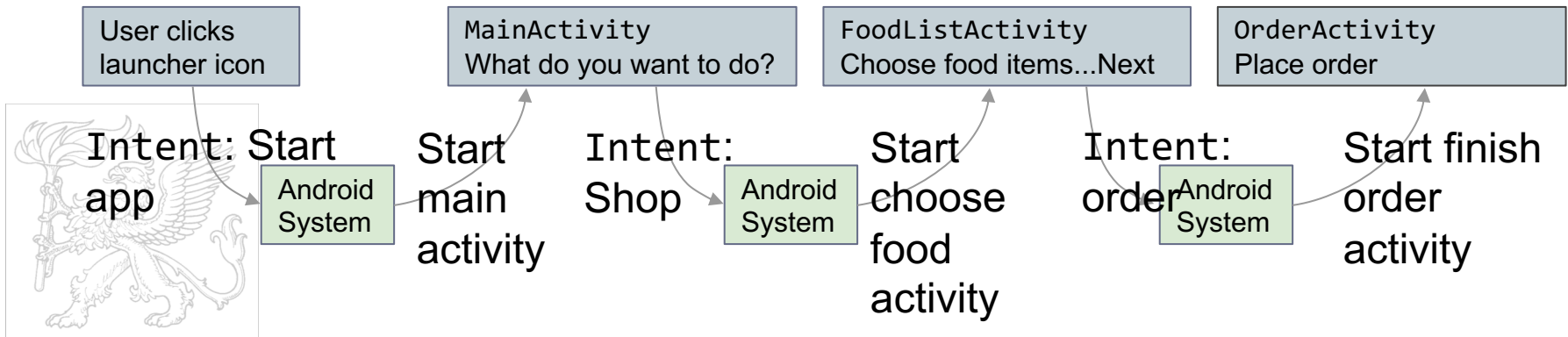
Telefonszám felhívása

```
Uri uri = Uri.parse("tel:8005551234");  
Intent it = new Intent(Intent.ACTION_DIAL, uri);  
startActivity(it);
```



Hogyan futnak az Activity-k

- ▶ Az Android futási környezeti kezeli az élelciklusukat
- ▶ Intent-tel indulnak (egy üzenettel)



Adatok küldése és fogadása

- ▶ Adat: egy információegység, URI-val megadható
- ▶ Extrák: egy – vagy több információ egység egy Bundle-ban név-érték párként

```
// Web oldal URL
```

```
intent.setData(  
    Uri.parse("http://www.google.com"));
```

```
// a Minta fájl URI
```

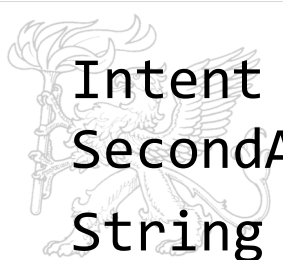
```
intent.setData(  
    Uri.fromFile(new  
    File("/sdcard/minta.jpg")));
```



Adatok küldése és fogadása

- `putExtra(String name, int value)`
⇒ `intent.putExtra("level", 406);`
- `putExtra(String name, String[] value)`
⇒ `String[] foodList = {"Rice", "Beans", "Fruit"};`
`intent.putExtra("food", foodList);`
- `putExtras(bundle);`
⇒ amennyiben sok az adata akkor először adjuk át a bundle-t

```
public static final String EXTRA_MESSAGE_KEY =  
    "com.example.android.twoactivities.extra.MESSAGE";
```



```
Intent intent = new Intent(this,  
    SecondActivity.class);  
String message = "Hello Activity!";  
intent.putExtra(EXTRA_MESSAGE_KEY, message);  
startActivity(intent);
```

Adatok elkérése

- `getData();`
⇒ `Uri locationUri = intent.getData();`
- `int getIntExtra (String name, int defaultValue)`
⇒ `int level = intent.getIntExtra("level", 0);`
- `Bundle bundle = intent.getExtras();`
⇒ Egy bundle-ként az összes adatot elkérjük.



Adat visszaadása az indító Activity-hez

```
public static final int CHOOSE_FOOD_REQUEST = 1;

Intent intent = new Intent(this, ChooseFoodItemsActivity.class);
startActivityForResult(intent, CHOOSE_FOOD_REQUEST);

// Create an intent
Intent replyIntent = new Intent();

// Put the data to return into the extra
replyIntent.putExtra(EXTRA_REPLY, reply);

// Set the activity's result to RESULT_OK
setResult(RESULT_OK, replyIntent);

// Finish the current activity
finish();
```



onActivityResult()

```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == TEXT_REQUEST) { // Identify activity
        if (resultCode == RESULT_OK) { // Activity succeeded
            String reply =
                data.getStringExtra(SecondActivity.EXTRA_REPLY);
            // ... do something with the data
        }
    }
}
```



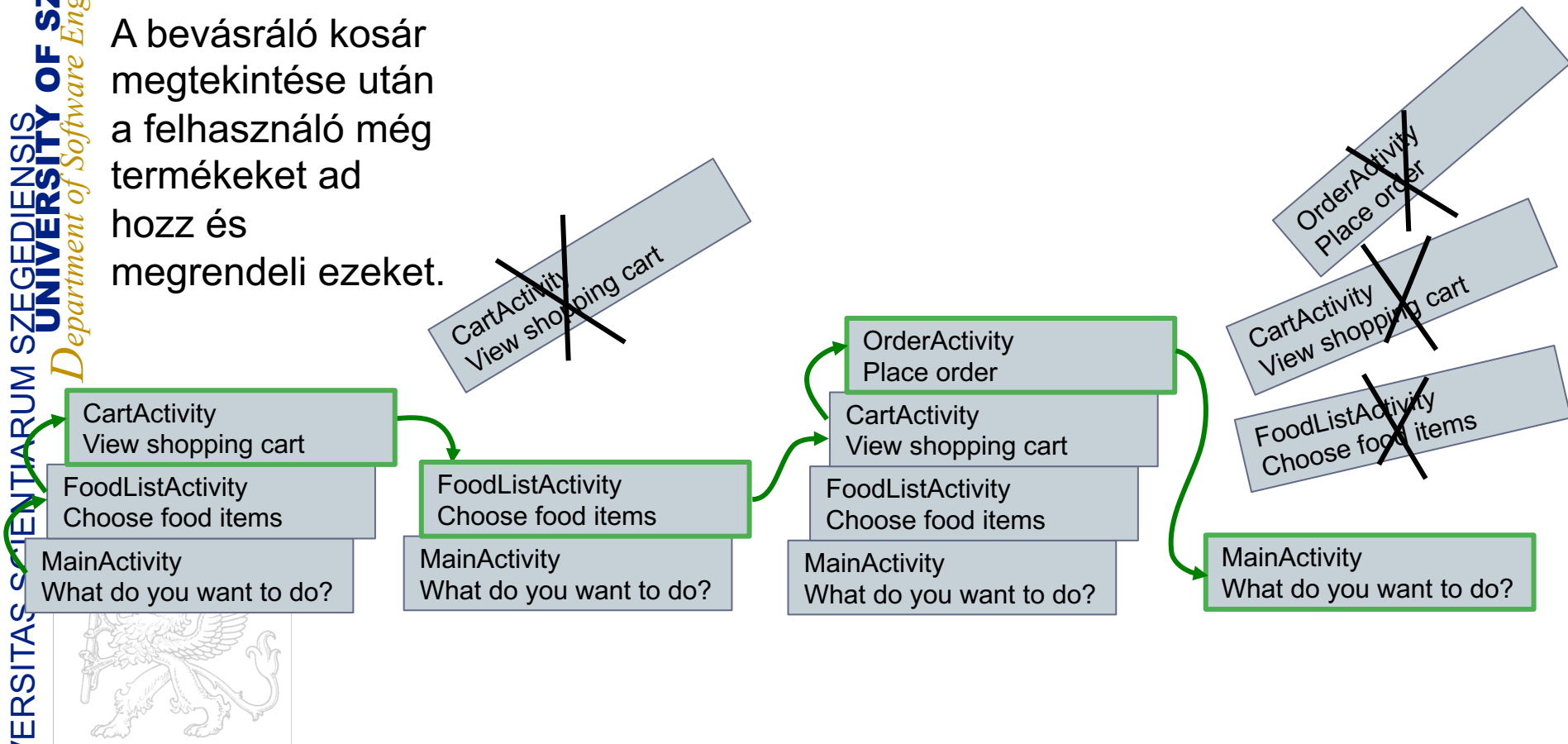
Navigáció

- ▶ Amikor egy Activity elindul akkor az előző leáll és bekerül az Activity vissza verembe
- ▶ Utolsónak-be elsőnek ki verem - amikor a mostani Activity befejeződik akkor az előző Activity-t elindítják



Activity verem

A bevásárló kosár megtekintése után a felhasználó még termékeket ad hozzá és megrendeli ezeket.



Két típusú navigáció

Ideiglenes vagy vissza navigáció

- az eszköz vissza gomjával
- Az Android vissza verme kezeli

Ős vagy fel navigáció

- az alkalmazás Fel gombja az akció menüben
- szülő- gyermek reláció segítségével adjuk meg az Andorid leíróban



Vissza navigáció

- ▶ Megőrzi a legutoljára nézett képrnyők sorrendjét
- ▶ Minden aktivitás megvan amit a mostani előtt elindított
- ▶ Minden feladatnak megvan a saját vissza verme
- ▶ A feladatok közötti váltás az adott feladatok vissza vermét aktiválja



Fel navigáció

- ▶ Az adott Activity szülőjéhez megy
- ▶ Az Android leíróban van megadva

```
<activity
  android:name=".ShowDinnerActivity"
  android:parentActivityName=".MainA
ctivity" >
</activity>
```



Összefoglaló

- ▶ UI elrendezések és erőforrások
 - Nézet, Nézet csoport, nézet hierarchia
 - Elrendezés szerkesztő, ConstraintLayout
 - Eseménykezelés
 - Erőforrások és mérésük
- ▶ Szöveg és gördülő nézetek
- ▶ Activity-k és intent-ek

