BUTE Department of Automation and Applied Informatics
AMORG
APPLIED MOBILE RESEARCH GROUP

Android alapú szoftverfejlesztés

Felhasználói felület készítése XML-alapokon

Segédlet

Forrás és további információk:

http://developer.android.com/guide/topics/ui/declaring-layout.html
http://developer.android.com/reference/android/view/View.html

Tartalom

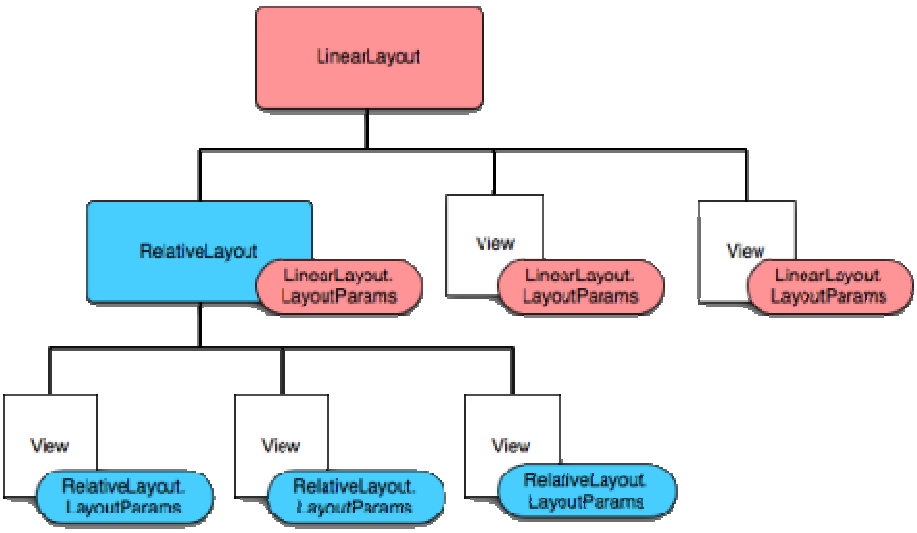
1 Alapfogalmak

Android platformon Activity-jeink felhasználói felületét View és ViewGroup objektumok egymásba ágyazott hierarchiája adja. A hierarchia deklarálása két alapvető módszer létezik:

- **View objektumok példányosítása kódból**, hierarchia összeállítása futásidőben.
- **Hiearchia deklarálása XML alapokon**, Activity-khez rendelése kódból, futásidőben.

A fenti két módszer küzül alapvetően ez utóbbi használata ajánlott, hiszen ezáltal lehetővé válik az alkalmazás felhasználói felületének elkülönítése az alkalmazáslogikától, mivel a felület leíró XML külön szerkeszthető, megjeleníthető.

A felület leíró XML-ben egy adott *node* a felület egy **View** (pl. TextView, ImageView) vagy **ViewGroup** elemének (pl. LinearLayout) felel meg. Az egyes elemek viselkedése, kinézete XML *attribútumok* megadásával szabályozható.



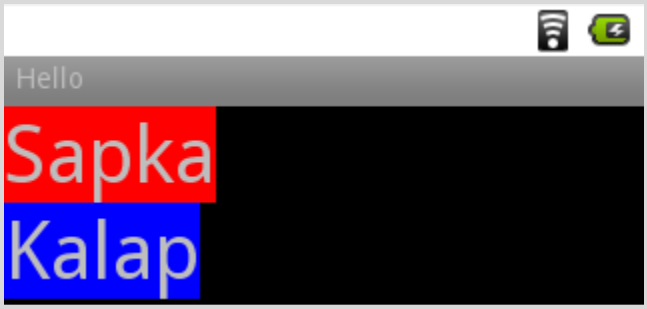
2 A LinearLayout elrendezés használata

A ViewGroup-ok nagyrészére igaz, hogy feladatuk gyermek View elemeik valamilyen **elrendezési stratégia (Layout)** szerinti megjelenítése.

Az egyik legegyszerűbb elrendezési stratégiát a **LinearLayout** valósítja meg. Feladata, hogy gyermekeit egymás után, sor- vagy oszlopfolytonosan rendezze el. Az, hogy az elemek sorban, vagy oszlopban helyezkedjenek el, az `orientation` attribútummal szabályozható (default érték a sor). Ennek működése a következő két példán megfigyelhető.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:text="Sapka"
        android:textSize="40dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:text="Kalap"
        android:textSize="40dp" />

</LinearLayout>
```

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:text="Sapka"
        android:textSize="40dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:text="Kalap"
        android:textSize="40dp" />

</LinearLayout>
```



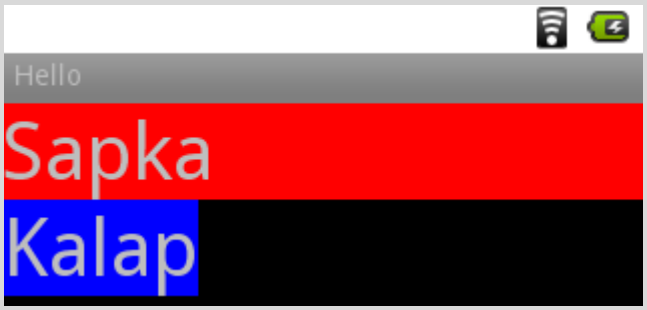
3 Alapvető layout attribútumok használata

3.1 Méretezés, a layout_width és layout_height paraméterek

Az egyik legtöbbet használt attribútumpáros a **View-k méretének megadására** használható `layout_width` és `layout_height`. A felbontás- és képernyőméret-függetlenség biztosítása érdekében **nem célszerű konkrét értékeket megadni** méretként. Ehelyett használhatjuk a két alapvető konstanst:

- `fill_parent` (=`match_parent`): A View az adott dimenzió mentén szülő elemét teljesen kitölti.
- `wrap_content`: Az adott dimenzió mentén a View mérete éppen akkora lesz, mint amekkorát tartalma igényel (pl. egy TextView éppen akkora lesz, hogy a megjelenített szöveg beleférjen).

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:text="Sapka"
        android:textSize="40dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:text="Kalap"
        android:textSize="40dp" />

</LinearLayout>
```



3.2 Súlyozás, a layout_weight paraméter

Tegyük fel, hogy a következő elrendezést szeretnénk leírni XML alapokon:

Figyeljük meg, hogy a középső TextView a teljes szabad helyet kitölti. Ennek szabályozására a `layout_weight` (súly) paraméter használható. Egy LinearLayout-on belül a View-k a **megadott súlyuk arányában fogják a szabad helyet kitölteni**:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="<"
        android:textSize="40dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:text="Hétfő"
        android:layout_weight="1"
        android:textSize="40dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=">"
        android:textSize="40dp" />

</LinearLayout>
```

Mivel a gyermekelemek közül egyedül a TextView kapott súlyt, a teljes maradék helyet lefoglalja a képernyő szélessége mentén.

A súlyozás egyéb layout „trükkök”-re is felhasználható, pl. három View között egyenlően felosztott sor leírására. A témakörről részleten a forrásban olvashatunk.

3.3 Igazítás, a layout_gravity és gravity paraméter

Egy View objektum (illetve tartalmának) elhelyezkedését a gravitáció paraméterekkel (`layout_gravity` és `gravity`) szabályozhatjuk. A számos előre megadott konstans (`center`, `left`, `right`, `center_vertical`, stb.) segítségével **a felület tartalma többféle módon pozícionálható**.

Az alapvető különbség a kétféle gravitáció között, hogy míg a `gravity` paraméter a **View tartalmát pozícionálja a View határát leíró téglalapon belül**, addig a `layout_gravity` a **View téglalapját pozícionálja a szülő Layout-on belül**. Az eltérő viselkedést a következő példa illusztrálja:

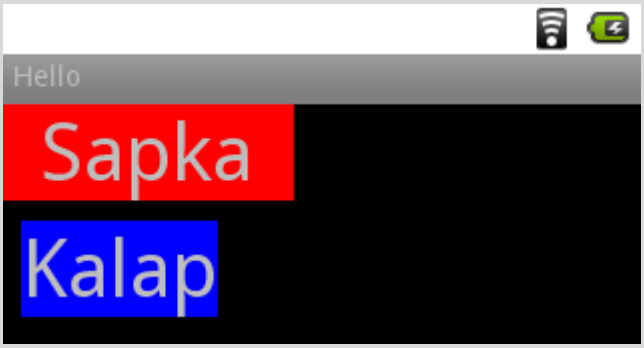
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:text="Sapka"
        android:gravity="center"
        android:textSize="40dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Kalap"
        android:layout_gravity="right"
        android:textSize="40dp" />

</LinearLayout>
```

3.4 A margin és a padding paraméterek

A **View objektumok közti szabad terület nagyságát** a `layout_margin` és a `padding` paraméterek szabályozzák. Megadásuknál a megismert mértékegységeket (dp, sp, stb.) használhatjuk. Mindkét paraméter definiálható a View összes határoló oldalára egyszerre (pl. padding=”10dp”), illetve megadható kiválasztott irányokban egyedi érték (pl. paddingLeft=”20dp”).

Az alapvető különbség a margin és a padding között, hogy a marginként megadott méret valóban „üres” terület lesz, míg **a padding területe a View része marad.** Ennek illusztrálására szolgál a lenti példa. Figyeljük meg, hogy a padding esetében a szabad terület is piros, hiszen az a TextView része.

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:text="Sapka"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        android:textSize="40dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:text="Kalap"
        android:layout_margin="10dp"
        android:textSize="40dp" />

</LinearLayout>
```



Egyes esetekben a padding és a margin közötti különbség fontos lehet (pl. a kattintás esemény vizsgálatakor a padding részre kattintva a View megkapja az eseményt).

4 Összetett felhasználói felület készítése

4.1 LinearLayout példa

Felmerül a kérdés, hogy miként írhatunk le a fenti példáknál összetettebb felhasználó felületeket. Az első módszer amit követhetünk, hogy **oszlopba és sorokba rendezett LinearLayout-ok egymásba ágyazásával** állítjuk elő a kívánt eredményt:

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

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="<"
            android:textSize="40dp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#00f"
            android:gravity="center"
            android:text="Hétfő"
            android:textSize="40dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=">"
            android:textSize="40dp" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:background="#f00"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <ImageView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:padding="10dip"
            android:src="@android:drawable/ic_dialog_info" />

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginRight="10dip"
            android:background="#00f"
            android:gravity="right"
            android:text="Nincs teendő."
            android:textSize="20dp" />

    </LinearLayout>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Hozzáadás"
        android:textSize="20dp" />

</LinearLayout>
```

Bár a fenti módszer előnye, hogy a fejlesztőtől kizárólag a viszonylag egyszerűen használható LinearLayout ismeretét követeli meg, hátránya, hogy túl összetett layout-ok esetében a többszintű egymásba ágyazás a feleslegesen sok objektum miatt pazarló megoldáshoz, nagyobb memória foglaláshoz, lassabb megjelenítéshez vezethet (pl. sok elemű lista megjelenítése scrollozás közben lassú lesz).
http://developer.android.com/resources/articles/layout-tricks-efficiency.html

Kétszintű LinearLayout használata az esetek többségében még elfogadható, ám ennél összetettebb hierarchiák leírására használjunk inkább RelativeLayout-ot.

4.2 A RelativeLayout használata

A RelativeLayout elrendezési stratégia használatával a View objektumok pozícióját egymáshoz, és a szülő ViewGroup-hoz viszonyítva adhatjuk meg. A megközelítés előnye, hogy **összetett kapcsolatokat írhatunk le egyszerűen és hatékonyan**, többszintű layout hierarchia használata nélkül.

A RelativeLayout gyermek elemeinek pozícionálására számos különböző layout_* paraméter használható:

- `layout_alignParent*`: A View relatív igazítása a szülőhöz (pl. alignParentLeft=”true” a View-t a RelativeLayout bal oldalához igazítja).
- `layout_toLeftOf`, `layout_toRightOf`: A View-t egy másik megadott View bal/jobb szélére igazítja.
- `layout_above`, `layout_below`: A View-t egy másik megadott View fölé/alá igazítja.
- stb. Részletes lista: http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html

A kapcsolatok leírásánál (azaz a fenti XML attribútumok értékének megadásakor) a korábban megismert `id` attribútumot használhatjuk. Pl.:

```
...
    <TextView
        android:id="@+id/first"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Első" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/first"
        android:text="Második" />
...
```

4.3 RelativeLayout példa

A RelativeLayout használatáról leírtakat a következő felület példa illusztrálja:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#00f"
    android:padding="10dp" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginRight="20dp"
        android:layout_marginTop="10dp"
        android:src="@android:drawable/ic_dialog_alert" />

    <TextView
        android:id="@+id/first"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@id/icon"
        android:text="Fontos"
        android:textSize="40dp" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/first"
        android:layout_toRightOf="@id/icon"
        android:background="#f00"
        android:text="Új teendő érkezett."
        android:textSize="20dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="Megtekint" />

</RelativeLayout>
```


Hello
Fontos
Megtekint
Új teendő érkezett.