XML-ben kell (lehet csak programból is) megadni, hogy a képernyőn milyen elemek, widgetek, stb. legyenek felrakva és milyen elrendezésben!

Elrendezés, layout

Négyféle van, egyébként 6.

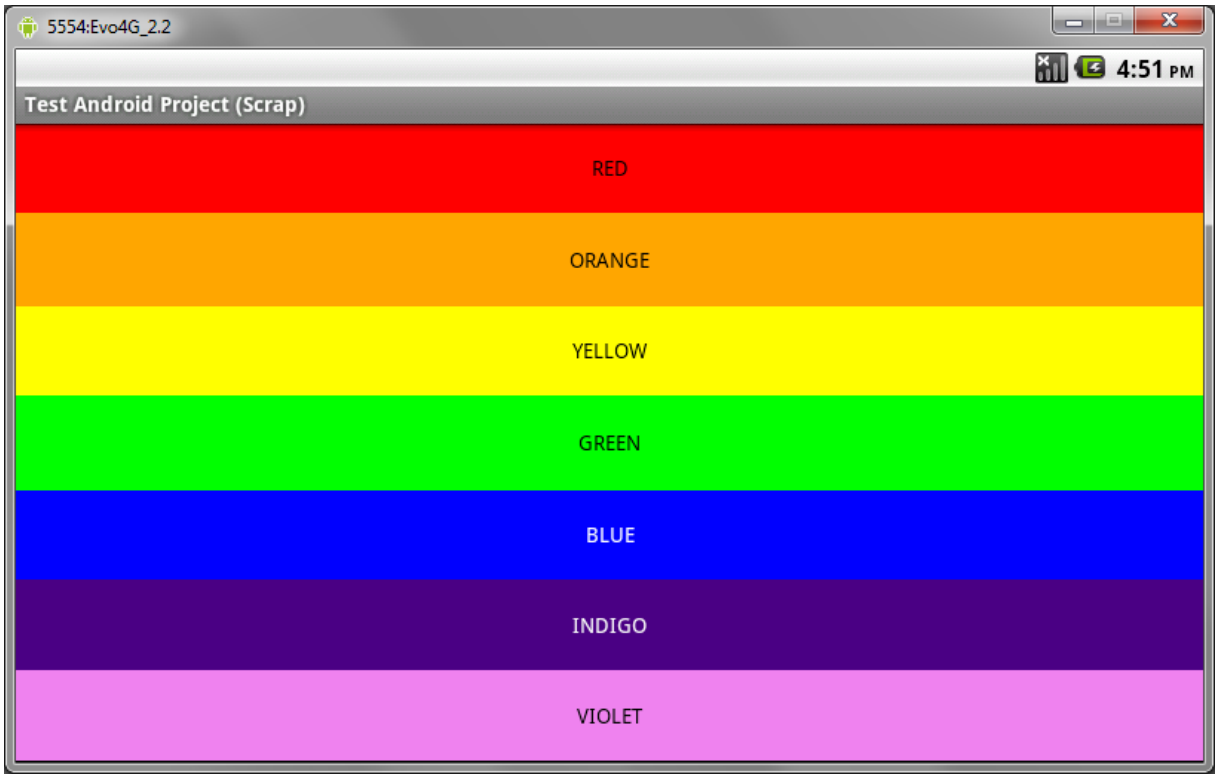
Linear, Relatív, List, Grid, Frame és Table. Azaz 6.

Az ötödik meg a Table. XML-ben kell megadni.

Nézzünk példát lineárisra:

```
<?xml version="1.0" encoding="utf-8"?>  //ezt ide beírod, és kész.
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"  // töltse ki a „szülőt”. azaz a képernyőt
    android:layout_height="fill_parent" //függőlegesen is.
    android:paddingLeft="16dp" //bal margó megadása
    android:paddingRight="16dp"// jobb margó. De nem kötelező!!!
    android:orientation="vertical" >  //egymás alá pakolja a widget-eket!
    <EditText  //ez egy szövegdoboz
        android:layout_width="fill_parent"  //vízszintesen érjen végig
        android:layout_height="wrap_content" // függlegesen a tartamhoz igazodjon
        android:hint="@string/to" />  //sugó
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Nézzünk egy másik példát!




```
<?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">
    <TextView android:text="RED" android:id="@+id/TextView01"
        android:layout_height="wrap_content" android:background="#f00"
        android:layout_width="fill_parent" android:layout_weight=".14"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="ORANGE" android:id="@+id/TextView02"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".15" android:background="#ffa500"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="YELLOW" android:id="@+id/TextView03"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#ffff00"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="GREEN" android:id="@+id/TextView04"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".15" android:background="#0f0" android:gravity="center"
        android:textColor="#000"></TextView>
    <TextView android:text="BLUE" android:id="@+id/TextView05"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#00f" android:gravity="center"
        android:textColor="#fff"></TextView>
    <TextView android:text="INDIGO" android:id="@+id/TextView06"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#4b0082"
        android:gravity="center" android:textColor="#fff"></TextView>
    <TextView android:text="VIOLET" android:id="@+id/TextView07"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#ee82ee"
        android:gravity="center" android:textColor="#000"></TextView>
</LinearLayout>
```

Nem nehéz, nem kell aggódni. Csupa címke (TextView) van felpakolva. Persze az android: textColor="#000" utasításból látod, hogy színezve vannak. Ami érdekes:

```
<TextView android:text="RED" android:id="@+id/TextView01"
        android:layout_height="wrap_content" android:background="#f00"
```

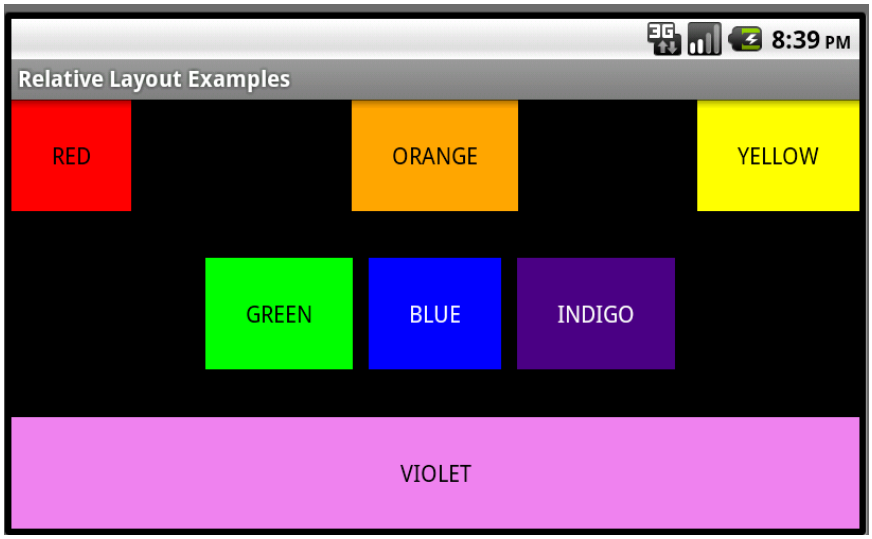
```
        android:layout_width="fill_parent" android:layout_weight=".14"
        android:gravity="center" android:textColor="#000"></TextView>
```

A text mondja meg a feliratot. Az id adja meg a nevét, amivel lehet azonosítani. A background a színt. A gravity azt mondja meg, hogy a szöveg hová kerüljön. És persze a végén le van zárva. Figyeld meg, ebben a példában máshogyan zárja le az XML utasításokat.

Ami érdekesebb a relatív layout.

http://mobile.tutsplus.com/tutorials/android/android-layout/

Példa



Nézzük hogy lehetett a fenti elrendezést elérni:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent">
    <TextView
        android:text="RED" // itt semmi sincs megadva. Alapból felülre, balra teszi
        android:id="@+id/TextView01"
        android:layout_height="wrap_content"
        android:background="#f00"
        android:gravity="center"
        android:textColor="#000"
        android:layout_width="wrap_content"
        android:padding="25dp"></TextView>  // a padding a margó
    <TextView
        android:text="ORANGE"
        android:layout_height="wrap_content"
        android:background="#ffa500"
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_centerHorizontal="true" //Azt mondja, h vízszintesen középre!!!
        android:padding="25dp"></TextView>
    <TextView
        android:text="YELLOW"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:gravity="center"
```

```
        android:textColor="#000"
        android:id="@+id/TextView03"
        android:layout_width="wrap_content"
        android:layout_alignParentRight="true"  //A szülő (azaz a képernyő) jobbszélére
teszi. Más nem kell, mert rakja folyamatosan ki
        android:padding="25dp"></TextView>
    <TextView
        android:text="GREEN"
        android:layout_height="wrap_content"
        android:background="#0f0"
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView04"
        android:layout_width="wrap_content"
        android:layout_toLeftOf="@+id/TextView05" == az 5-ös Viewnak a balszélére teszi.
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView> //és persze függőlegesen középre
    <TextView
        android:text="BLUE"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView05"
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="10dp"
        android:padding="25dp"></TextView>
    <TextView
        android:text="INDIGO"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView06"
        android:layout_width="wrap_content"
        android:layout_toRightOf="@+id/TextView05"
        android:background="#4b0082"
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView>
    <TextView
        android:text="VIOLET"
        android:layout_height="wrap_content"
        android:background="#ee82ee"
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView07"
        android:layout_alignParentBottom="true"
        android:layout_width="fill_parent"
        android:padding="25dp"></TextView>
</RelativeLayout>
```

Ha nem vizuálisan csinálod, akkor figyelj arra, hogy az Eclipse felajánlja a lehetőséget!
További lehetőségek a neten….

Az xml-ben megadott elemek java programban történő használata:

Így kell beállítani azt, h milyen elrendezést akarsz:

```
setContentView(R.layout.activity_main);
```

Az XML-ben lévő objektumokat azonosítani kell a programban. A megoldás:
Példa gombra:
Deklarálsz egy gomb objektumot:

```
private Button doitbutton;
```

Majd az xml-ben lévő gombot így tudod megtalálni:

```
doitbutton = (Button) findViewById(R.id.Run);
```

Innentől kezdve az XML-ben lévő *Run* gombot doitbutton-ként tudod használni.

Vagy:
Van egy textView1 az XML-ben.
Akkor deklarálni kell egyet:

```
private TextView myText;

myText=(TextView)findViewById(R.id.textView1);
```

Eseménykezelés – gombra kattintás

A gombra kattintást kétféleképpen kezelhetjük. Az egyik módszer szerint minden gombhoz definiálni kell az eseménykezelőt. Például a doitbutton esetében:

```
doitbutton.setOnClickListener(new View.OnClickListener() {

                    public void onClick(View v) {
                          //ide beírod, h mit csináljon
                          // pl. zárja be az alkalmazást!
                          finish();
                          System.exit(0);



                    }
        });
```

Ha sok gombod van, akkor elég problémás lehet pl. 11x ilyet leírni. A megoldás a következő:

Implementálnod kell az eseménykezelő interfészét. Azaz:

```
public class MainActivity extends Activity implements OnClickListener{
```

Az Activity után oda kell írnod: *implements OnClickListener.* Azaz implementálnod kell, fel kell fűznöd az eseményfigyelők közé a kattintás esemény kezelőjét.

Majd meg kell írnod az eseménykezelő, az onClick-et:

```
public void onClick(View v) {

  Toast pieceToast = Toast.makeText(getApplicationContext(),
toastText.getText(), Toast.LENGTH_SHORT);

pieceToast.show();

}

showToast.setOnClickListener(this);
```

Azaz, a showToast nevű gomb eseménykezelőjét aktiváljuk így.

De honnan tudja, hogy mire kattintottam?

Ha több gomb van, akkor tudni kellene, hogy melyik gombra kattintottak.

Az alábbi példában rengeteg gomb van. Íme:

```
public class Szamologep extends Activity implements OnClickListener{

     Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16;

b1=(Button)findViewById(R.id.button1);

    b2=(Button)findViewById(R.id.button2);

    b3=(Button)findViewById(R.id.button3);

    b4=(Button)findViewById(R.id.button4);

…
```

Majd:

```
b1.setOnClickListener(this);

    b2.setOnClickListener(this);

    b3.setOnClickListener(this);

    b4.setOnClickListener(this);
```

A gombok az XML-es leírásból származnak. Tehát 16 gombunk van. Ehhez nem írunk külön eseménykezelőt, hanem 1-et.

```
public void onClick(View v) {
     switch(v.getId())
     {
        case R.id.button1:e1.setText(e1.getText()+"9"); break;
        case R.id.button2:e1.setText(e1.getText()+"8"); break;
        case R.id.button3:e1.setText(e1.getText()+"7"); break;
        case R.id.button4:e1.setText(e1.getText()+"6"); break;
        case R.id.button5:e1.setText(e1.getText()+"5"); break;
        case R.id.button6:e1.setText(e1.getText()+"4"); break;
        case R.id.button7:e1.setText(e1.getText()+"3"); break;
        case R.id.button8:e1.setText(e1.getText()+"2"); break;
        case R.id.button9:e1.setText(e1.getText()+"1"); break;
        case R.id.button10:e1.setText(e1.getText()+"."); break;
        case R.id.button11:e1.setText(e1.getText()+"0"); break;
        case R.id.button12:e1.setText(e1.getText().subSequence(0,
e1.getText().length()-1)); break;
        case R.id.button13:e1.setText(e1.getText()+"+"); break;
        case R.id.button14:e1.setText(e1.getText()+"-"); break;
```

```
        case R.id.button15:e1.setText(e1.getText()+"*"); break;
        case R.id.button16:e1.setText(e1.getText()+"/"); break;
        case R.id.button17:e1.setText(e1.getText()+"="); break;
     }

}
```

Tehát a v.getId() adja vissza, hogy melyik gombra kattintottunk. Ha a button1-re, akkor ezt csinálja. Ha a másikra, akkor azt, stb.

Tehát a második megoldás lépései:

Az activity deklarálásakor *implements OnClickListener*{

Majd a gombok definiálása után:

```
b1.setOnClickListener(this);
```

Be kell állítani a gomb eseménykezelőjét.

Majd az utolsó lépésben megírni az onClick metódust.

```
public void onClick(View v) {
}
```

Persze az importban ott kell szerepeltetni a

```
import android.view.View.OnClickListener;
```

sort is, de ez a CTRL-SHIFT-O megnyomásakor is hozzáíródik.

Nézzük meg a radiobutton használatát!

Az xml-ben definiálni kell:

```
<RadioGroup
        android:id="@+id/radioNoise"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_alignParentRight="true"
         android:layout_toRightOf="@id/textView6"
         android:layout_below="@id/textView00">

        <RadioButton
            android:id="@+id/saltpapper"
             android:layout_width="wrap_content"
         android:height="15dp"
            android:checked="true"
            android:textSize="12dip"
            android:button="@drawable/button_radio"
            android:text="Salt and pepper noise">
         </RadioButton>

        <RadioButton
```

```
            android:id="@+id/noise"
            android:layout_width="wrap_content"
          android:height="15dp"
              android:textSize="12dip"
              android:button="@drawable/button_radio"
            android:text="Speckel noise">
           </RadioButton>

    </RadioGroup>
```

Majd a java programban:

```
private RadioGroup noise;
private RadioButton noisetype;
```

Hivatkozás:

```
noise = (RadioGroup) findViewById(R.id.radioNoise);
```

Ennek nem onClick-je van, hanem CheckedChange-je.

```
  noise.setOnCheckedChangeListener(this);
```

A fentiből látszik, hogy megint implementálni kell az eseménykezelőt, azaz az osztály fejlécében:

```
….implements OnClickListener, OnCheckedChangeListener {..
```

És már csak meg kell írni az eseménykezelőt:

```
public void onCheckedChanged(RadioGroup group, int checkedId) {
          switch(checkedId){
               case R.id.saltpapper:
imageflag=0;setData();message.setText("'Salt and pepper' noise");break;
               case R.id.noise :
imageflag=1;setData();message.setText("Speckle noise");break;
          }

     }
```

Toast használata

Egy felbukkanó popup ablak. Kb. olyan, mint a MessageBox.Show();
A makeText metódussal jön létre. Ennek három argumentuma van:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();

Lehet persze egyben is:
Toast.makeText(context, text, duration).show();


Vagy egyben:
```

```
Toast pieceToast = Toast.makeText(getApplicationContext(), toastText.getText(),
Toast.LENGTH_SHORT);

pieceToast.show();
```

// van LENGTH_LONG is.

Alapból a képernyő alján, középre igazítva jelenik meg. De ezen lehet változtatni.

Használni kell a setGravity(int, int, int) metódust. Három paramétere van: Gravity constant, an x-position offset, and a y-position offset.

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

A jobb felső sarokba teszi.

Listview használata

Tegyünk egy listView elemet a képernyőre!

```
<ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="37dp" >
    </ListView>
```

Írjuk meg a programot!

```
private ListView Lista;
      private ArrayAdapter<String> listAdapter1;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            Lista = (ListView) findViewById( R.id.listView1 );
            //mi legyenek benne
            String[] szovegek = new String[] { "Homogenous image", "Image with noise", "Image
with noise and edge", "Image with horizontal edge", "Image with vertical edge"};
            ArrayList<String> szovegList = new ArrayList<String>();
        szovegList.addAll( Arrays.asList(szovegek) );
           listAdapter1 = new ArrayAdapter<String>(this, R.layout.simplerow, szovegList); //a
simplrerow egy xml fájl, leírjs a sorokat


            // Set the ArrayAdapter as the ListView's adapter.
            Lista.setAdapter( listAdapter1 );
            //eseménykezelője
            Lista.setOnItemClickListener(new OnItemClickListener() {
                  @Override
                  public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                    Toast.makeText(getApplicationContext(),
                      "Click ListItem Number " + position, Toast.LENGTH_LONG)
                      .show(); // a position adja vissza, h melyiket választották, ki,
kattintottak rá
```

```
                }
            });

    }
```

Az Adapter egyfajta híd az AdapterView és a z adatok között. Létrehozunk egy String tömböt az listview elemeivel. Létrehozunk egy ArrayList-et, azaz egy tömböt, amelybe belerakjuk a korábbi String tömböt, azaz a szöveget. Majd az Arrayadapter-t kell létrehozni, és hozzárendelni a korábbi tömblistát! Elég körülményes szerintem is. És már csak az eseménykezelőt kell megírni!

SMS küldése

Rövid szöveges üzenetek programozott küldésére a platform két módszert is biztosít a fejlesztők számára. Vagy implicit Intent-t állítunk össze, és a rendszerre bízzuk az SMS tényleges küldését, vagy pedig mi magunk kezeljük a küldéssel kapcsolatos életciklus-eseményeket.

Implicit Intent használata

A rendszernek küldött Intentnek a következő beállításokat kell tartalmaznia:

Akció: Intent.ACTION_SENDTO

Adat: "sms:[címzett telefonszáma]': Uri-ként

Extrák: "sms_body" kulccsal az üzenet szövege

```
Engedélyeztetni kell a Manifest-ben az üzenet küldését!
<uses-permission android:name="android.permission.SEND_SMS" />
```

A program:

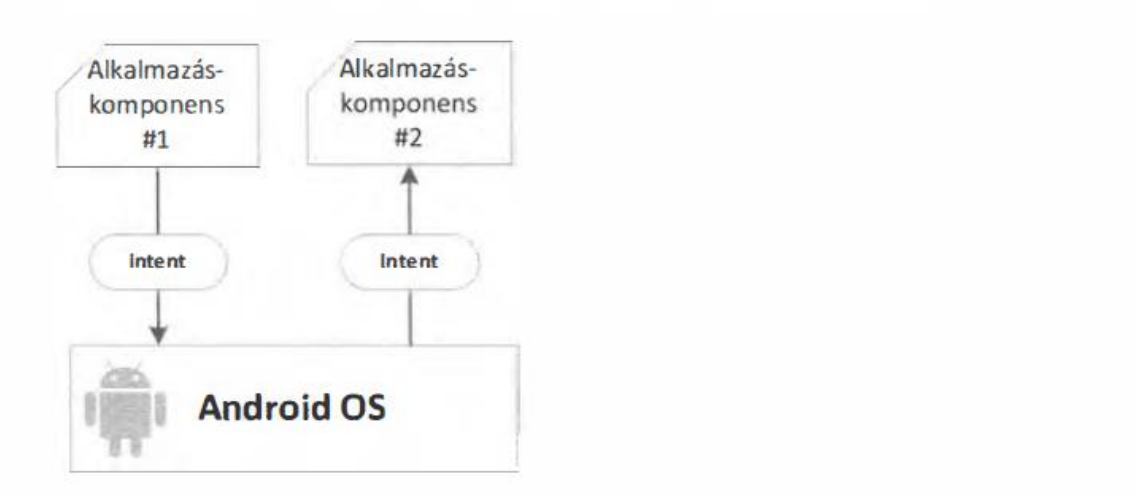
```
Button buttonSend;
      EditText textPhoneNo;
      EditText textSMS;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            buttonSend = (Button) findViewById(R.id.button1);
            textPhoneNo = (EditText) findViewById(R.id.editText1);
            textSMS = (EditText) findViewById(R.id.editText2);
            buttonSend.setOnClickListener(new OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                String phoneNo=textPhoneNo.getText().toString();
                      String sms = textSMS.getText().toString();


                      try {
                      SmsManager smsManager = SmsManager.getDefault();
                            smsManager.sendTextMessage(phoneNo, null, sms, null,
null);
                            Toast.makeText(getApplicationContext(), "SMS Sent!",
                                        Toast.LENGTH_LONG).show();
                      } catch (Exception e) {
                            Toast.makeText(getApplicationContext(),
                                        "SMS faild, please try again later!",
                                        Toast.LENGTH_LONG).show();
                            e.printStackTrace();
                      }




                }
          });
```

Az Intent fogalma

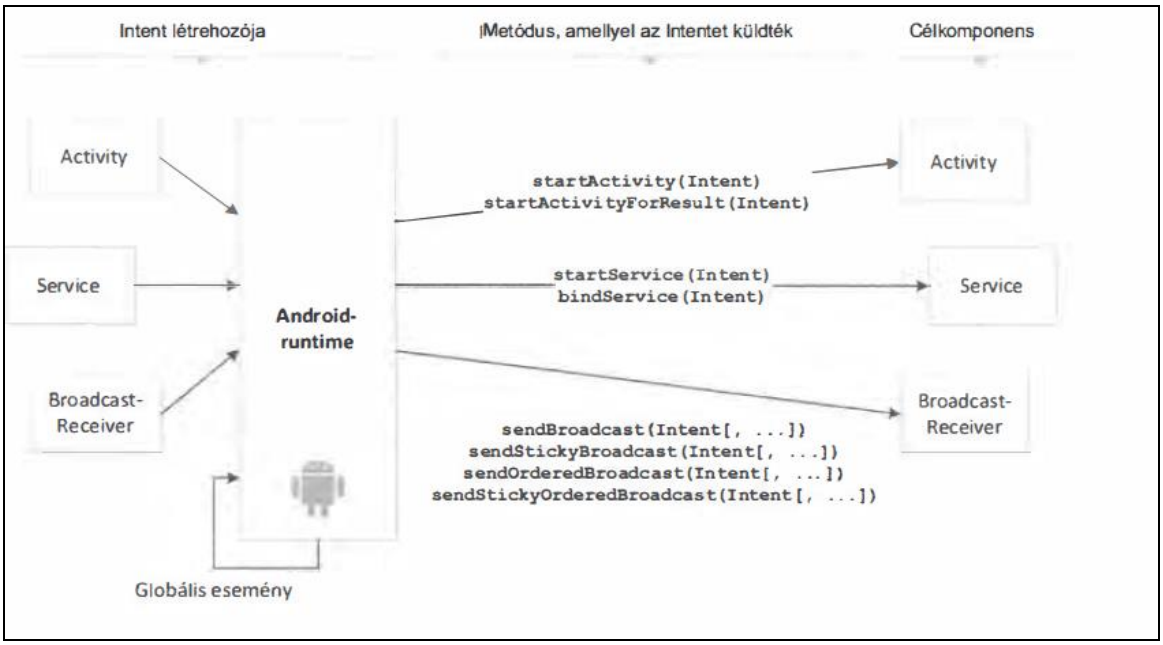
Android operációs rendszer esetén az alkalmazások háromféle komponensbó1 állhatnak, ezek a felhasználói felülettel rendelkező Activity, a háttérbeli futásra képes Service, valamint a rendszerszintű események kezelésére képes BroadcastReceiver. Egy alkalmazás tipikusan több különböző típusú komponenst tartalmaz. Nézzünk meg egy igen egyszerű, jegyzetkészítő Android-alkalmazást, amely lehetőséget biztosít a felhasználónak új feljegyzések létrehozására és a meglévők listázására, szerkesztésére, törlésére. Ennek legkézenfekvőbb implementációja két Activityt foglal magában, amelyek közül a nyitóképernyő-komponens a jegyzetek listázására, a másik pedig azok szerkesztésére vagy új létrehozására szolgál. Már az ilyen egyszerű esetekben is szükség van a komponensek közti kommunikációra, hiszen egy feljegyzés szerkesztésekor a nyitóképernyőn lévő listát megjelenítő Activitynek el kell indítania a módosításért felelős

társát, és tudatnia kell vele, hogy a felhasználó melyik jegyzetet választotta ki. Android platforrnon az alkalmazáskomponensek adatcseréjének elsődleges módszere az Intentek használata. Ez nem más, mint egy Java-osztály, amelyre passzív adatstruktúraként tekintünk. Késői, futásidejű kötést valósít meg az alkalmazáskomponensek között. Az adat, amelyet hordoz, és így maga az objektum is mindig valamilyen esemény absztrakt leírására szolgál. Ez lehet elvárt esemény, amelyet az Intent hatására szeretnénk előidézni. Jellemzően Activity vagy Service indítása, BroadcastReceiver regisztrálása). Ha a rendszerben broadcast üzenet keletkezik, akár az operációs rendszer által (például kimenő hívás, akkumulátorfeszültség alacsony stb.), akár egy alkalmazás kezdeményezésére (például egy e-mail applikáció rendszerszintű értesítést küldhet új levél érkezésekor), akkor az üzenet szintén Intent-objektumként van jelen a rendszerben, és jut el az arra feliratkozott komponensekhez. Ekkor tehát egy bekövetkezett esemény attribútumait tartalmazza. Bármilyen okból is jött létre, az Intent sohasem közvetlenül adódik át a komponensek között, minden esetben az operációs rendszeren keresztül történik a kézbesítése. Vagy maga az Android generálja valamilyen rendszeresemény hatására, vagy pedig a futtató környezet kapja egy komponenstó1, és gondoskodik a célba juttatásáról (lásd az alábbi ábrát).



Az Intent-objektum az Androidon a szabványos háromféle alkalmazáskomponens közti kommunikáció eszköze, továbbá a rendszer is ennek segítségével szolgáltat információkat a globális események bekövetkezéséró1. Az Intent eredete lehet tehát Activity, Service, BroadcastReceiver vagy maga az operációs rendszer. A potenciális célpontok szintén ugyanezek a komponenstípusok, ám egy bizonyos módon a runtime-hoz került Intent mindig egy meghatározott osztály példányánál köthet ki, mégpedig a következó1mél:

- Csak Activity lehet a fogadó objektum, ha bármilyen komponens startActivity() vagy startActivityForResult() metódussal küldte a rendszernek.
- Service lesz a célpont, hogyha az Intent célja szolgáltatás indítása a startService() metódussal vagy futásidejű kötés megvalósítása már futó szolgáltatáshoz a bindService() hívás hatására.
- A BroadcastReceiver az egyetlen osztály, amely képes fogadni és kezelni a sendBroadcast() metódus segítségével elküldött Intentet.



Bárhol jött is létre az Intent-objektum, az Android megkeresi a megfelelő címzettet, szükség esetén példányosítja és elindítja, valamint átadja neki a híváshoz tartozó és az aktiválását kiváltó adatcsomagot. Az Intent-koncepció egyik legnagyobb erőssége és egyben a platformok közti különlegessége is abban rejlik, hogy a megcélzott komponens nemcsak ugyanazon a processzen belül lehet, ahonnan az Interrtet küldték, hanem más alkalmazás valamelyik modulja is címezhető ilyen módon. Ebbó1 természetesen az is következik, hogy saját applikációnkat is felkészíthetjük máshonnan küldött kérések fogadására és kiszolgálására, így az Android operációs rendszerre telepített alkalmazások egy nagy, lazán csatolt szolgáltatáshálózatot alkothatnak. A platform ezzel a mechanizmussal teremti meg a lehetőséget a gyári, előre telepített szoftverek lecserélhetőségére, hiszen egy harmadik féltól származó applikáció is képes lehet kiszolgálni bármilyen akciót igénylő Intentet. Nézzük meg az Interrtek alapvető jellemzőit és lehetőségeit!

4. 2.1ntent felépitése

Az Intent nem más, mint egy olyan információcsomag, amely egy megtörtént vagy elvárt esemény leírását tartalmazza. Felhasználási módja sokrétű, így attribútumai is többféle kombinációban lehetnek kitöltve vagy üresen hagyva attól függően, hogy mit ír le, vagy éppen mire szeretnénk használni

Mezői között megtalálható a címzett komponens neve, a bekövetkezett vagy végrehajtandó akció azonosítója, az adat, amelyen ez értelmezve van, de tartalmazhat még egyedi kulcs-érték párokat (extrák), további megkötéseket a címzett komponenssel kapcsolatban vagy flageket, ha új Activity indítását kezdeményezzük az Intent segítségével. Ezeket az attribútumokat (lásd a következő ábrát) mindig olyan kombinációban kell kitöltenünk, amely az aktuális feladathoz vagy eseményhez szükséges, az API nem írja elő kötelezően egyik feltöltését sem. Nézzük meg a mezők részletes bemutatását. (2-öt legalábbis.)

Akció
Egy sztringkonstans az elvárt vagy broadcast hatására küldött Intent esetén a megtörtént eseményt jelzi. A platform rengeteg előre definiált, beépített akciót tartalmaz, de az alkalmazások egyedi konstansokat is definiálhatnak. Az operációs rendszer által használt akciók teljes listája folyamatosan bővül az új verziók kiadásával, a mindenkori legfrissebb kollekció mindig elérhető az API-referenciában, az Intent-osztály leírásánál. A fontosabb, leggyakrabban használt akciókat a következő táblázat mutatja be.

Konstans	Célpont típusa	Leírás
ACTION_VIEW	Activity	Az Intent adatmezőjében lévő entitás megnyitása olvasásra (például fájl, névjegy)
ACTION_EDIT	Activity	Az Intent adatmezőjében lévő entitás megnyitása szerkesztésre
ACTION_PICK	Activity	Választás az adatmezőben lévő URI által hivatkozott listából (például partner választása a névjegyzékből)
ACTION_SEND	Activity	Az adatmező tartalmának megosztása más alkalmazással
ACTION_CALL	Activity	Az adatmezőben átadott telefonszám felhívása
ACTION_BATTERY_LOW	Broadcast-Receiver	Akkufeszültség alacsony
ACTION_BOOT_COMPLETED	Broadcast-Receiver	A telefon bekapcsolt (ennek kezelésével lehet automatikusan indítani az alkalmazásunkat)
ACTION_POWER_CONNECTED	Broadcast-Receiver	Töltőre került az eszköz (ilyenkor érdemes például szinkroznizálni a szerverrel, a cache-t karbantartani)

Fejlesztéskor mindig ellenőriznünk kell, hogy az alkalmazásunk által lekezelt akció az Android melyik verziójától számítva létezik. Ha ez magasabb, mint az alkalmazás által megkövetelt minimális API-szint, akkor programozóként fel kell készülnünk arra, hogy nem minden eszközön lesz elérhető
ez a bizonyos akció. Az akciómező nemcsak az esemény azonosítására szolgál, hanem meghatározza
a kérés feldolgozására képes komponens típusát (Activity vagy BroadcastReceiver), valamint az Intent további- leginkább az adat és az extrák- attribútumait is erősen befolyásolja, mint ahogy egy metódus signatúrája rögzíti annak nevén kívül a paramétereit és a visszatérési érték típusát is. Az akció konstans beállítására a setActio n(), lekérésére a getActio n() metódusok szolgálnak.

Adat

Ha az Intent által jelzett - megtörtént vagy igényelt - esemény önmagában nem értelmezhető, akkor a kapcsolódó adat URI-ja (Universal Resource Identifier, általános erőforrás-azonosító) és MIME-típusa14 szintén része az üzenetcsomagnak Az akció meghatározhatja a mellette lévő adat típusát, hiszen például ACTION_CALL esetén csak "tel:/!" kezdetű telefonszám, míg ACTJON_EDJT beállításakor például dokumentum, névjegy vagy fénykép átadásának van értelme. A MIME-típus beállításának célja triviális, csak olyan komponens dolgozhatja fel az Intentet, amely képes az URI által meghatározott típus ú adat megfelelő kezelésére. Ez különösen igaz az általános jelentésű
akciók esetében, hiszen egészen más feladatot jelent egy szöveges állományt és egy mp3-as fájlt szerkesztésre megnyitni. Persze az URI-ból gyakran egyértelműen következik a fájl típusa (pl. pdf, doc, jpg), a platform mégis biztosít lehetőséget annak a "kézi" beállítására, amelyet az Intent későbbi helyes kezelésének érdekében ajánlott is kihasználni.
Az adatmező beállítására a setData(), a setType() és a setDataAndType(), míg lekérdezésére a getData() és a get Type() metódusok állnak rendelkezésre.

Komponensnév

Annak a kom ponensnek a neve, amelynek kezelnie kell az Intentet, az üzenet explicit címzettje. Típusát tekintve ComponentName objektum, ez egyrészt a célkomponens minősített osztálynevébó1 (fully qualified class name, például "teszt.projekt.app.SzovegSzerkesztoActivity") és annak az alkalmazásnak a csomagnevébó1 (package name, például "teszt.projekt") áll, amelyben a komponens
megtalálható. Szükség van a csomagnév megadására, hiszen az nem feltétlenül egyezik meg a minősített osztálynév elejével.

És akkor jönne a grafika…