

ListView - przykład widoku listowego

Rozpoczynamy edycję projektu

Skoro projekt już mamy, na pierwszy ogień weźmiemy przygotowanie layoutu. W tym celu najprościej z zakładki **Composite** przeciągnąć na nasz layout komponent o nazwie **ListView** a następnie rozciągnąć go wedle własnego życzenia, u mnie to będzie 100% powierzchni operacyjnej. Poniżej kod naszego pliku XML zarządzającego układem komponentów.

```
<RelativeLayout 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"
    tools:context=".List" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" >

    </ListView>

</RelativeLayout>
```

Zwróćmy tu uwagę na **id** naszego ListView, będzie nam potrzebne do przechwycenia referencji komponentu z poziomu kodu Javy.

Definiujemy kod XML naszego pojedynczego elementu listy

W tym celu w folderze layout tworzymy nowy plik XML i definiujemy tam wygląd pojedynczego wiersza listy. U mnie to będzie najprostszy TextView

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/Row"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="15sp" >

</TextView>
```

Zaprogramowanie całości - operacje na Activity

Przyszł czas, aby połączyć wszystko w jedną działającą całość.

```

package pl.listviewexample;

import java.util.ArrayList;
import java.util.Arrays;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class List extends Activity {

    private ListView list ;
    private ArrayAdapter<String> adapter ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list);

        list = (ListView) findViewById(R.id.listView1);

        String cars[] = {"Mercedes", "Fiat", "Ferrari", "Aston Martin", "Lamborghini", "Skoda", "Volkswagen"}

        ArrayList<String> carL = new ArrayList<String>();
        carL.addAll( Arrays.asList(cars) );

        adapter = new ArrayAdapter<String>(this, R.layout.row, carL);

        list.setAdapter(adapter);
    }
}

```

Teraz po krótko opiszę co ciekawsze linijki w powyższej klasie.

****Linia 11 -****Nasza klasa musi rozszerzać klasę Activity (W bardziej zaawansowanym przykładzie, byłyby to już ListActivity)

Linia 12,13- Tworzymy obiekty widoku listowego i adaptera. Adapter jest to taki byt, który łączy nasz ListView z danymi mającymi być prezentowane jako lista.

****Linia 21 -****Przechwytyjemy referencję do naszego widoku listowego

****Linia 23 -****Tworzymy tablicę przechowującą stringi. Powinno się to zrobić troszkę inaczej, za pomocą strings.xml, ale przykład jest na tyle trywialny, że posługujemy się trywialnymi mechanizmami :-)

Linia 25,26 - Tworzymy obiekt [listy tablicowej](#) i przypisujemy mu naszą tablicę z samochodami

****Linia 28 -****Konfigurujemy stworzony w 13 linijce obiekt adaptera. Jako parametry podajemy mu Activity, adres pliku zawierającego definicję pojedynczego elementu i nazwę naszego obiektu listy tablicowej.

****Linia 30 -****Łączymy nasz widok listowy z naszym adapterem

Powyższy przykład powinien w skrócie wyjaśnić jak korzystać z komponentu ListView. W następnym artykule postaram się opisać tworzenie bardziej zaawansowanego widoku listowego, z własnym adapterem.