



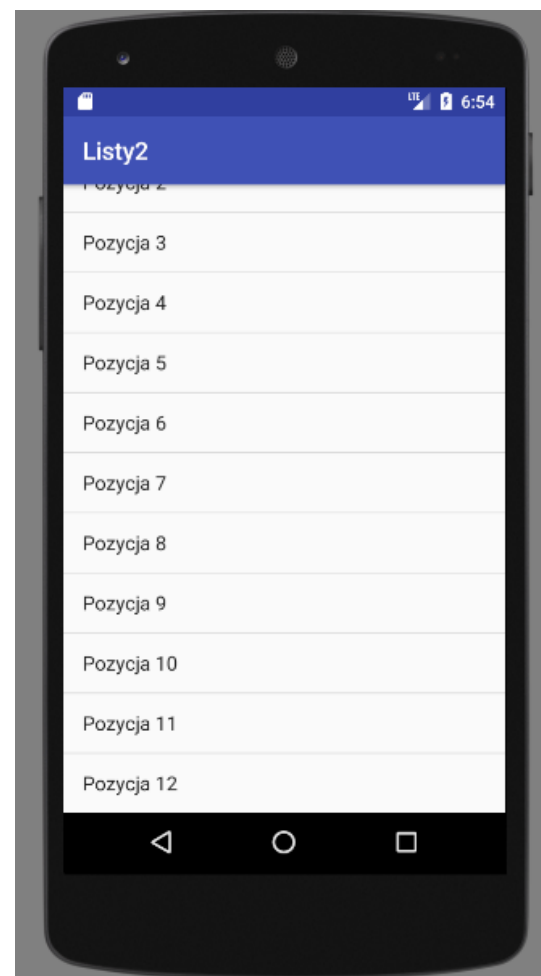
APLIKACJE MOBILNE

Wykład 10

dr Artur Bartoszewski

Widok listy

Komponent **ListView** odpowiada za wyświetlenie dowolnych elementów w postaci listy. Kiedy elementów jest więcej niż może pomieścić ekran, tworzy się pasek przewijania.



1. W pliku XML dodajemy kontrolkę ListView

```
9      <ListView
10          android:id="@+id/lista"
11          android:layout_width="wrap_content"
12          android:layout_height="wrap_content">
13      </ListView>
```

2. W pliku .java - w metodzie onCreate widoku:

```
ListView l1 = (ListView) findViewById(R.id.lista);  
String[] tab = new String[] {"Pozycja 1", "Pozycja 2", "Pozycja 3", "Pozycja 4",  
                             "Pozycja 5", "Pozycja 6", "Pozycja 7", "Pozycja 8",  
                             "Pozycja 9", "Pozycja 10", "Pozycja 11", "Pozycja 12",};  
final ArrayAdapter<String> adapter =  
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, tab);  
l1.setAdapter(adapter);
```

1. Znajdujemy uchwyt do ListView.
2. Przygotowujemy tablicę wartości do wyświetlenia (tablicę można także stworzyć w zasobach).
3. Tworzymy ArrayAdapter, który dopasuje nam tablicę do komponentu.
4. Dodajemy adapter do komponentu.

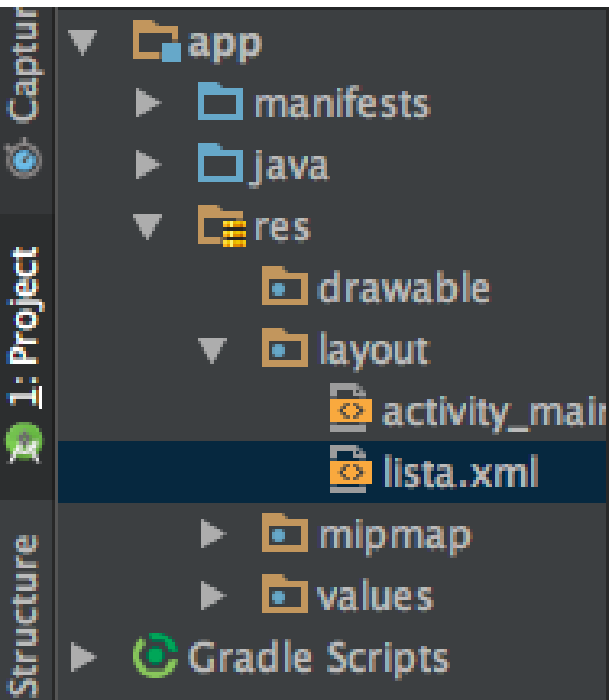
3. Reakcja listy na kliknięcie w element

```
AdapterView.OnItemClickListener clListener =  
    new AdapterView.OnItemClickListener() {  
        @Override  
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l)  
        {  
            Toast.makeText(MainActivity.this, String.valueOf(i),  
                           Toast.LENGTH_SHORT).show();  
        }  
    };  
l1.setOnItemClickListener(clListener);
```

Tworzymy `OnItemClickListener` o nazwie `clListener` (nazwa własna).

Indeks pozycji na którą kliknięto znajduje się w zmiennej „i”

3. Własny layout listy



Pracę rozpoczynamy od stworzenia pliku a folderze res/layout

3. Własny layout listy

```
TextView
1  <?xml version="1.0" encoding="utf-8"?>
2  <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3      android:id="@+id/Element"
4      android:layout_width="fill_parent"
5      android:layout_height="wrap_content"
6      android:padding="10dp"
7      android:textSize="20sp"
8      android:textColor="#ac2911"
9      android:textStyle="italic">
10 </TextView>
```

Tworzymy plik layout-u zawierający widok <TextView />

Nie wypełniamy widoku tekstem. Ustawienie innych opcji – według uznania

Inna wersja przygotowania listy elementów

```
resources
<resources>
  <string name="app_name">Listav02</string>

  <string-array name="lista_elementow">
    <item>Pozycja 1</item>
    <item>Pozycja 2</item>
    <item>Pozycja 3</item>
    <item>Pozycja 4</item>
    <item>Pozycja 5</item>
    <item>Pozycja 6</item>
    <item>Pozycja 7</item>
    <item>Pozycja 8</item>
    <item>Pozycja 9</item>
    <item>Pozycja 10</item>
    <item>Pozycja 11</item>
    <item>Pozycja 12</item>
  </string-array>
</resources>
```

Zamiast tablicy stringów zdefiniowanej w pliku .java możemy taką tablicę umieścić w pliku **strings**

Inna wersja przygotowania listy elementów

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(
    this,
    R.layout.lista,
    getResources().getStringArray(R.array.lista_elementow));
lista.setAdapter(adapter);
```

Aby skorzystać z tak przygotowanej tablicy posługujemy się funkcją **`getResources().getStringArray()`**

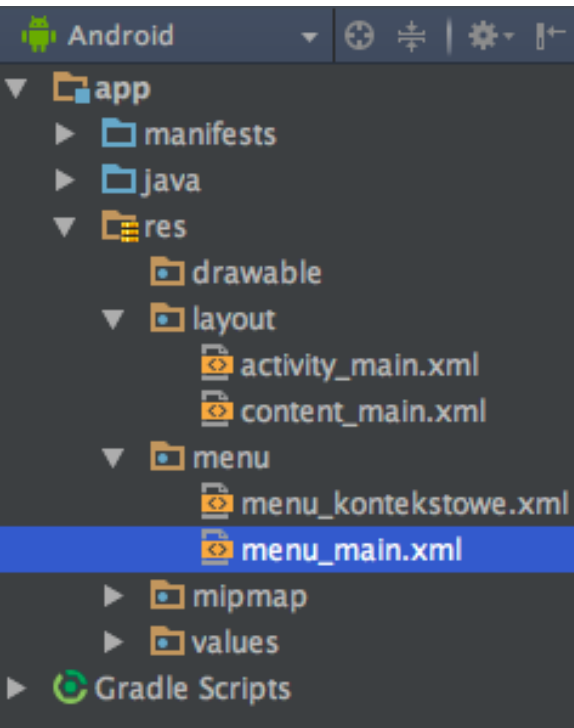
Menu opcji

Standardowe menu opcji w Androidzie



Aplikacje mobilne

I – przygotowanie zawartości menu



W folderze res tworzymy folder menu. W nim dodajemy plik main_menu.xml

```
menu item
<menu 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"
    tools:context="pl.uniwersytetradom.bartoszewski.artur.menu01.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never"/>
    <item
        android:id="@+id/zrob_cos_01"
        android:orderInCategory="101"
        android:title="Opcja pierwsza"
        app:showAsAction="never"/>
    <item
        android:id="@+id/zrob_cos_02"
        android:orderInCategory="102"
        android:title="Opcja druga"
        app:showAsAction="never" />
    <item
        android:id="@+id/zrob_cos_03"
        android:orderInCategory="103"
        android:title="Opcja trzecia"
        app:showAsAction="never" />
</menu>
```

[I18N] Hardcoded string "Opcja pierwsza", should use @string resource [more...](#) (%F1)

Elementem głównym jest `<menu />`. Każdej pozycji odpowiada `<item />`.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

- W kodzie .java nadpisujemy funkcję **onCreateOptionsMenu(Menu menu)**.
- W funkcji tej umieszczamy obiekt **getMenuInflater().inflate()** jego zadaniem jest rozwinięcie layoutu menu , który otrzymał w parametrze.
- Na tym etapie menu wyświetla się, lecz jeszcze nic nie robi.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    TextView t1 = (TextView) findViewById(R.id.textView01);

    switch (id) {
        case R.id.action_settings: t1.setText("Wybrano settings"); return true;
        case R.id.zrob_cos_01: t1.setText("Wybrano opcje pierwsza"); return true;
        case R.id.zrob_cos_02: t1.setText("Wybrano opcje druga"); return true;
        case R.id.zrob_cos_03: t1.setText("Wybrano opcje trzecia"); return true;
    }

    return super.onOptionsItemSelected(item);
}
```

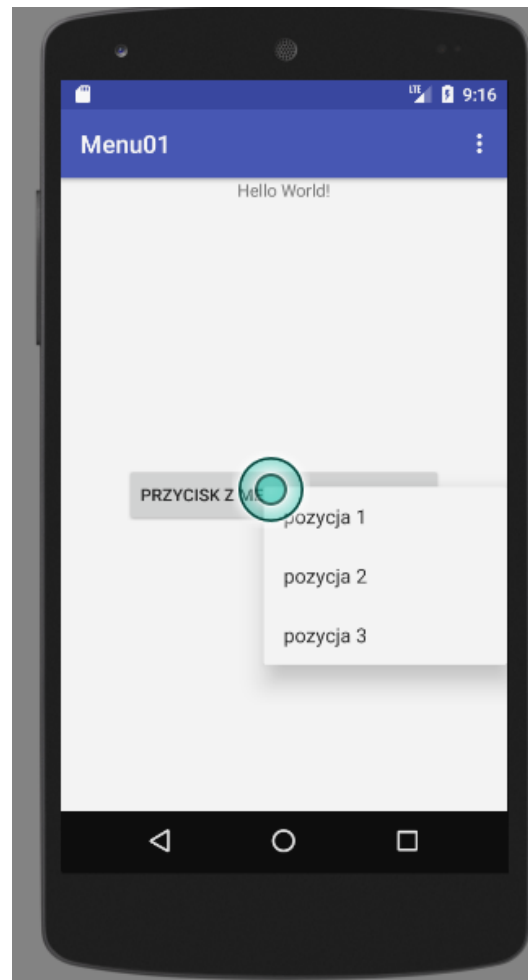
Kolejnym krokiem jest dodanie funkcji **onOptionsItemSelected(MenuItem item)**.

Funkcja otrzymuje w parametrze wskaźnik do elementu listy, który ją wywołał (item)

Menu kontekstowe

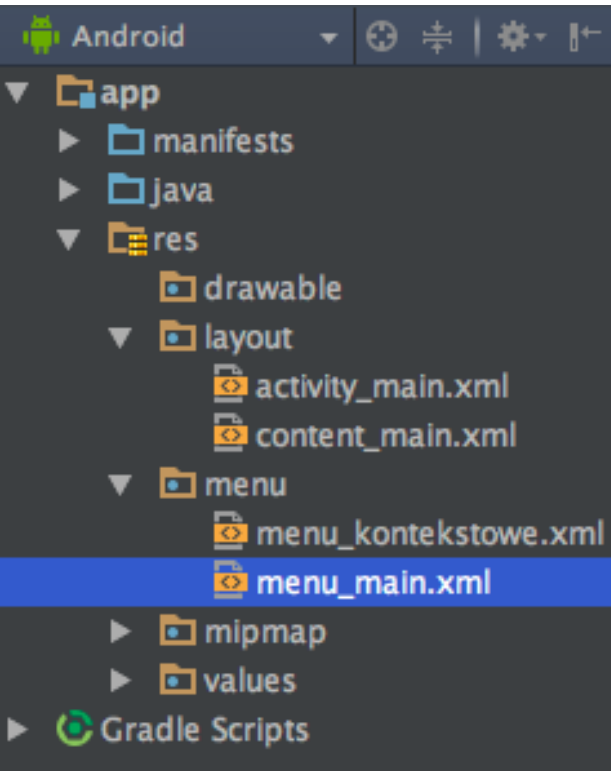
Menu pojawiające się po długim dotknięciu (przytrzymaniu) kontrolki.

Działa niezależnie od onClick oraz słuchacza kliknięć.



Aplikacje mobilne

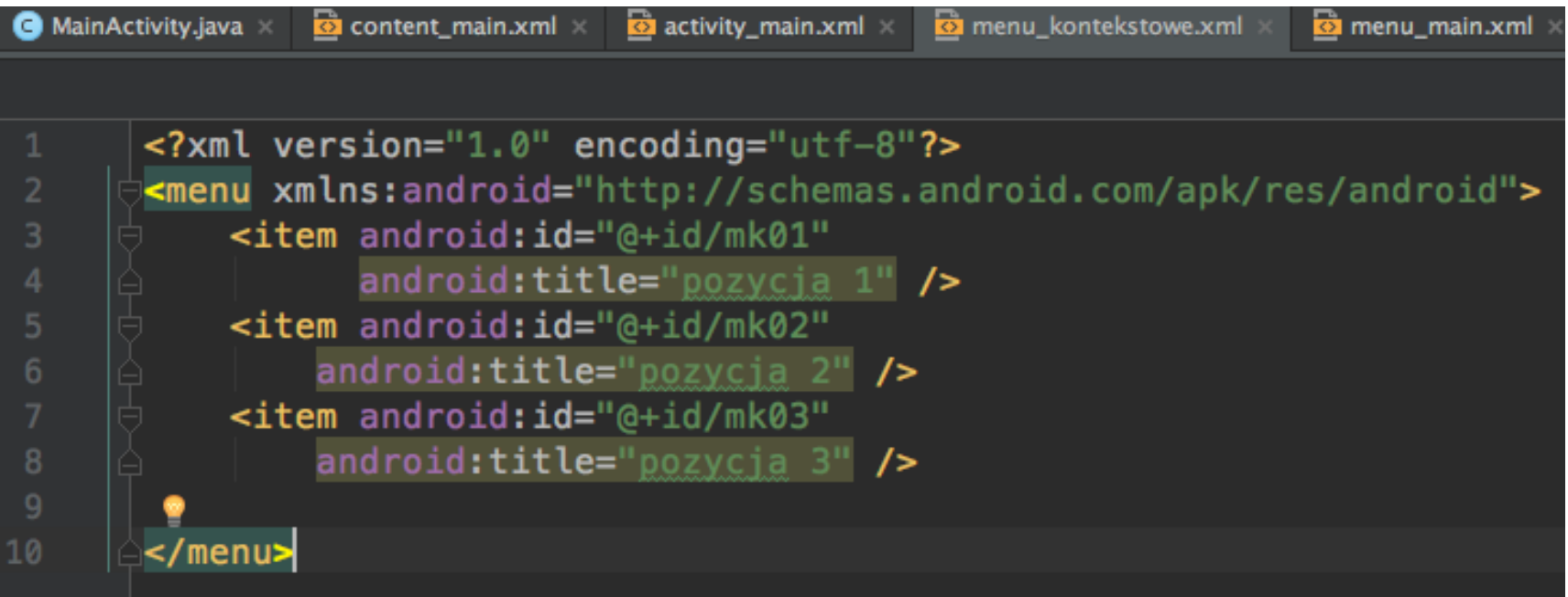
I – przygotowanie zawartości menu



W folderze res tworzymy folder menu.

W nim dodajemy **plik menu_kontekstowe.xml**
(nazwa własna)

II – przygotowanie zawartości menu



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3     <item android:id="@+id/mk01"
4         android:title="pozycja 1" />
5     <item android:id="@+id/mk02"
6         android:title="pozycja 2" />
7     <item android:id="@+id/mk03"
8         android:title="pozycja 3" />
9
10 </menu>
```

II – powiązanie menu z przyciskiem

```
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21          Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
22          setSupportActionBar(toolbar);
23
24          Button button = (Button) findViewById(R.id.button01);
25          registerForContextMenu(button);
26
27      }
```

- W **onCreate** odnajdujemy uchwyt do przycisku (lub innego elementu, któremu który chcemy wyposażyć w menu kontekstowe).
- Rejestrujemy menu – poleceniem **registerForContextMenu()** z parametrem, którym jest uchwyt do przycisku

III – Wyświetlenie menu

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.menu_kontekstowe, menu);
    super.onCreateContextMenu(menu, v, menuInfo);
}
```

- W kodzie .java nadpisujemy funkcję **onCreateContextMenu()**.
- W funkcji tej umieszczamy obiekt **getMenuInflater().inflate()** jego zadaniem jest rozwinięcie layoutu menu , który otrzymał w parametrze.
- Na tym etapie menu wyświetla się, lecz jeszcze nic nie robi.

IV – Obsługa zdarzenia kliknięcia

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    TextView t1 = (TextView) findViewById(R.id.textView01);

    switch (id) {
        case R.id.mk01: t1.setText("Wybrano opcje pierwsza z menu kontekstowego"); return true;
        case R.id.mk02: t1.setText("Wybrano opcje druga z menu kontekstowego"); return true;
        case R.id.mk03: t1.setText("Wybrano opcje trzecia z menu kontekstowego"); return true;
    }
    return super.onOptionsItemSelected(item);
}
```

- Kolejnym krokiem jest dodanie funkcji **onContextSelected(MenuItem item)**.
- Funkcja otrzymuje w parametrze wskaźnik do elementu listy, który ją wywołał (item)

