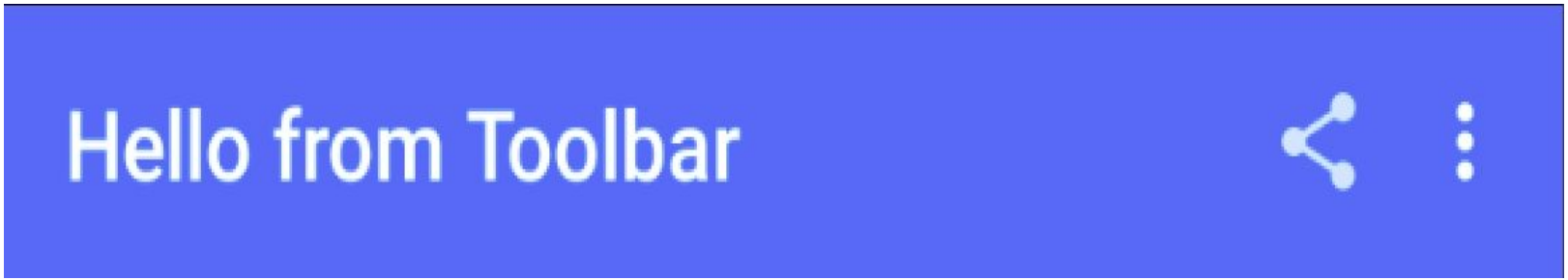
Android GUI/Kontrolki/Toolbar

Wojciech Bomba, Aleksander Ciołak, Łukasz Łukowicz

Co to jest Toolbar?



Jest to (typowo niebieski) pasek u góry naszej aplikacji umożliwiający **nawigację**, popularne **akcje** wykonywane w danym momencie przez użytkownika oraz **więcej** możliwości/akcji które zwykle kryją się pod **rozwijanym** menu oznaczonym trzema kropkami ustawionymi w pozycji pionowej (patrz rys.)

Tworzenie toolbara.

1. Zadbanie o zależności oraz SDK

```
dependencies {
  ...
  compile 'com.android.support:appcompat-v7:25.2.0'
}
```

ORAZ

```
android {
        defaultConfig {
                minSdkVersion 16
```

Tworzenie toolbara cd.

1. Usunięcie domyślnego toolbara jaki został wyprodukowany.

res -> values -> styles.xml ->

zamieniamy linijeczkę

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

na:

<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

Tworzymy nowy plik w którym będzie nasz toolbar (w folderze layout -> new -> layout resource file -> root ustawiamy na **android.support.v7.widget.Toolbar** ponadto wysokość oraz kolor tła ustawiamy jak niżej

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v7.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"

   android:layout_height="wrap_content"

   android:background="#DDD">

</android.support.v7.widget.Toolbar>
```

Tworzenie toolbara cd.

w activity_main.xml inkludujemy nasz toolbar.

PS ustalamy sobie id dowolne ale nazwa pliku w katalogu @layout musi sie zgadzać

```
<include

  android:id="@+id/app_bar"

  layout="@layout/app_bar"/>
```

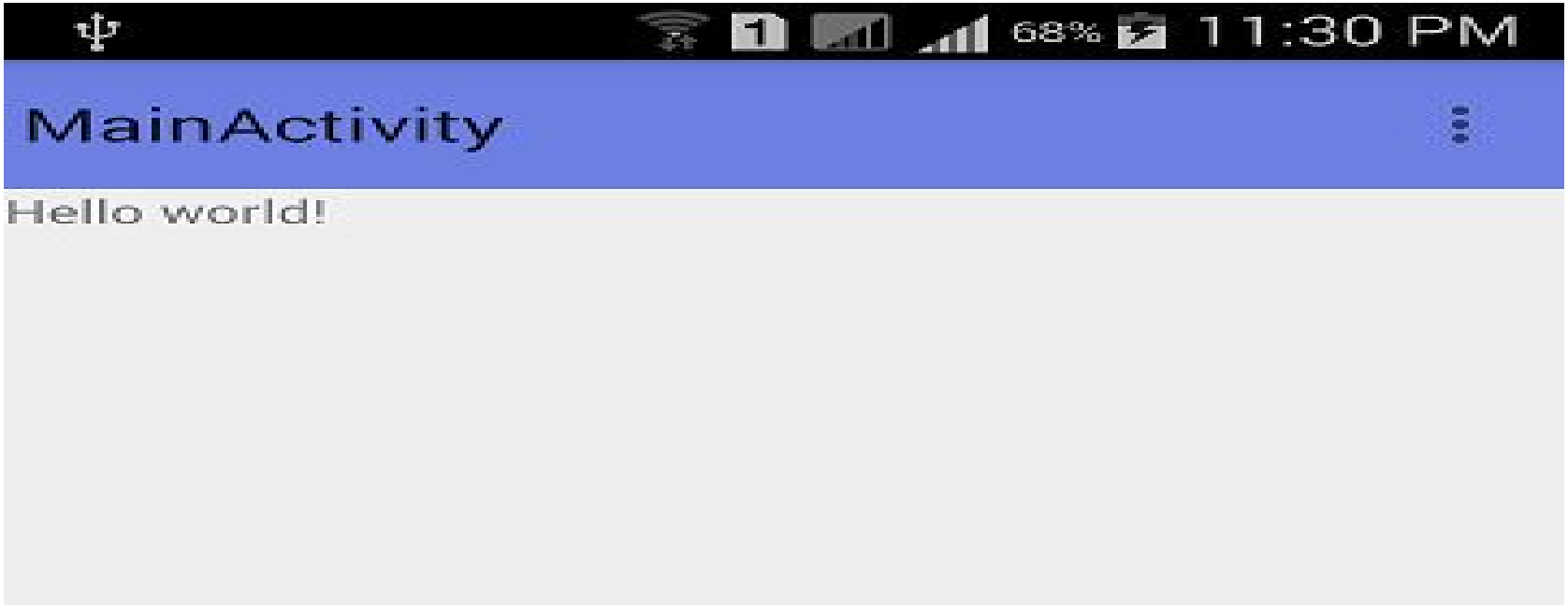
dodawanie toolbara w activity_main.xml

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Toolbar toolbar = (Toolbar) findViewById(R.id.app_bar);
   setSupportActionBar(toolbar);
}
```

dodawanie toolbara w activity_main.xml cd

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.action_settings:          return true
      default
         return super.onOptionsItemSelected(item);
   }
}
```

obecny stan



Dodawanie akcji (w folderze /res/menu/*.xml)

```
<item  android:id="@+id/message6"
  android:icon="@android:drawable/ic_dialog_email"
  android:title="wiadomosc6"
  app:showAsAction="ifRoom"/>
<item   android:id="@+id/action_settings"
  android:orderInCategory="100"
  android:title="@string/action_settings"
  app:showAsAction="never" />
```

NOWY DODANY ITEM

JEŚLI JEST MIEJSCE TO POKAŻ JAK NIE TO PRZENIEŚ DO MENU

ZAWSZE JEST TO POD ROZWIJANYM MENU

Obsługa akcji: metoda onOptionsItemSelected(MenuItem item)

```
int id = item.getItemId();

if (id == R.id.action_settings) {    return true;}

if (id == R.id.message) {

   Toast.makeText(getApplicationContext(), "wcisnieto message", Toast.LENGTH_SHORT).show();
return true;

}

return super.onOptionsItemSelected(item);
```

Dodawanie Buttona UP

Ustalanie relacji parent-child pomiędzy aktywnościami:

```
<!-- A child of the main activity -->
    <activity
        android:name="pl.kis.agh.edu.pl.example.MyChildActivity"
        android:label="@string/title_activity_child"
        android:parentActivityName="pl.kis.agh.edu.pl.example.MainActivity" >
```

Uruchamianie Buttona UP (metoda onCreate dziecka)

```
Toolbar myChildToolbar = (Toolbar) findViewById(R.id.my_child_toolbar);

    setSupportActionBar(myChildToolbar);

    ActionBar ab = getSupportActionBar();

    // Enable the Up button

    ab.setDisplayHomeAsUpEnabled(true);
```

Zadanie

Stwórz swój własny toolbar który zawiera rozwijane menu oraz co najmniej jedną akcję która bedzie się znajdowała poza tym menu (jej uruchomienie ma powodować dowolną widoczną zmianę jak zmiana tła albo wyświetlenie Toast).

#ProHint Stwórz nowy projekt z szablonu Basic Activity nie zaś Empty Project następnie usuń zbędny element na dole ekranu (zapamiętaj lokalizację ikonki koperty) -> dopisz odpowiedni item do menu.xml (jako ikonkę podaj zapamiętany adres ikonki koperty `android:icon="@android:drawable/????????`) -> obsłuż akcję w onOptionsItemSelected podając odpowiedni id (id podobnie jak ikonkę trzeba było podać)itema po czym np zmień kolor tła

Input Controls

Input Controls – grupa komponentów wśród których wyróżniamy takie kontrolki UI jak:

- Button

- Text Field (Edit Text, AutoCompleteTextView)

- CheckBox

- Radio Button (RadioGroup, RadioButton)

- Toogle Button

- Spinner

- Picker

Dodanie kontrolek odbywa się metodą „drag and drop” lub poprzez modyfikacje pliku XML naszego layoutu.

Edit Text

Kontrolka, która dopuszcza możliwość edycji pola. Edit Text definiujemy w następujący sposób :

```
<EditText
    android:id="@+id/phone"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/phone_hint"
    android:inputType="phone" />
```

Istnieją różne typy pól, każde z nich determinuje tryb pojawiającej się klawiatury po kliknięciu na takie pole, odbywa się to poprzez określenie właściwości android:inputType. Wśród wartości inputType wyróżniamy np. „phone” który determinuje tryb numerycznej klawiatury oraz „text” który pozwala na wpisanie wartości znakowych czy „password” który ukrywa wpisywany tekst. Właściwość android: hint to placeholder,który dostarcza informacje o polu. Istnieje możliwość określenia przycisku który ma się pojawić na klawiaturze jako zatwierdzenie edycji poprzez ustawienie właściwości pola imię Options pola np. „action send” lub „actionSearch”

```
android:imeOptions="actionSend"
```

Edit Text - obsługa eventu

```
editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            System.out.println("Action Send");
            handled = true;
        }
        return handled;
    }
});
```

Button

Istnieją 3 sposoby na zdefiniowanie buttona:

1.Button z textem

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

2. Button z ikona

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

3. Button z tekstem i ikona

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

Button - obsługa eventu

Istnieją 2 możliwości zdefiniowania akcji która ma być odpalona w odpowiedzi na event :

1 . poprzez określenie w pliku xml funkcji która ma być odpalona w momencie kliknięcia na przycisk i zaimplementowanie jej w activity

```
android:onClick="clickButton"
```

```
public void clickButton(View view) {
    System.out.println("You are clicked button");
}
```

2. druga metoda to zdefiniowanie dla danego przycisku listenera w klasie activity

```
imageButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        System.out.println("You are clicked imageButton");
    }
});
```

CheckBox

Komponenty, które są niezależne umożliwiają zaznaczenie bądź odznaczenie kontrolki, takie zdarzenia możemy obsłużyć.

Checkbox definiujemy w następujący sposób:

```
<CheckBox
    android:id="@+id/salami"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/salami" />
```

Checkbox - obsługa eventu

Podobnie jak w przypadku buttonów można zdefiniować odpowiedz na eventy na dwa sposoby:

1. Zdefiniowanie w pliku xml funkcji,która ma być odpalona w odpowiedzi na event

```
android:onClick="onCheckboxClicked"

public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch (view.getId()) {
        case R.id.kurczak:
            if (checked)
                System.out.println("Zazanaczyłes kurczak");
            else
                System.out.println("Odznaczyłes kurczak");
            break;
        case R.id.salami:
            if (checked)
                System.out.println("Zazanaczyłes salami");
            else
                System.out.println("Odznaczyłes salami");
            break;

    }
}
```

Checkbox - obsługa eventu

2. Zdefiniowanie listenera dla konkretnego checkboxa

```
CheckBox checkBoxSalami = (CheckBox) findViewById(R.id.salami);


checkBoxSalami.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        boolean checked = checkBoxSalami.isChecked();
        if (checked)
            System.out.println("Zaznaczyles salami");
        else
            System.out.println("Odznaczyles salami");
    }
});
```

Radio Buttons

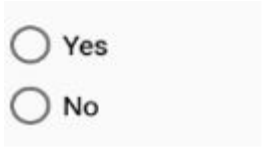
Radio Buttons to kontrolki, które najczęściej definiowane są w Radio Group pozwalają wówczas zaznaczyć tylko jedną z podanych opcji (tylko jeden Radio Button w obrębie Radio Group).

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

    <RadioButton
        android:id="@+id/radio_yes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="@string/yes" />

    <RadioButton
        android:id="@+id/radio_no"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="@string/no" />

</RadioGroup>
```



Radio Buttons - obsługa eventów

Definiowanie eventów odbywa się na 2 sposoby:

1. Sposób - określenie akcji która ma być odpalana w momencie zdarzenia.

```
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/radio_group"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/radio_yes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Yes" />

    <RadioButton
        android:id="@+id/radio_no"
        android:layout_width="180dp"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="No" />

</RadioGroup>
```

```
public void onRadioButtonClicked(View view) {

    switch (view.getId()) {
        case R.id.radio_yes:
                System.out.println("Zazanaczyłes yes");
            break;
        case R.id.radio_no:
                System.out.println("Zazanaczyłes no");
            break;

    }
}
```

Radio Buttons - obsługa eventów

2. Ustawienie listenera, który nasłuchuje na zmianę w obrębie zdefiniowanej Radio Group (składającej się z RadioButtonów)

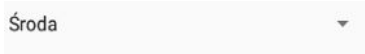
```
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.radio_group);
radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        radioButton = (RadioButton) findViewById(checkedId);
        System.out.println("Zaznaczyles " + radioButton.getText());

    }
});
```

Spinners

To komponent, który umożliwia wybranie jednej wartości z dropdowna

```
<Spinner
    android:id="@+id/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```



Zdefiniowanie zawartości spinnera definiujemy w pliku strings.xml

```
<string-array name="days">
    <item>Poniedziałek</item>
    <item>Wtorek</item>
    <item>Środa</item>
    <item>Czwartek</item>
    <item>Piatek</item>
    <item>Sobota</item>
    <item>Niedziela</item>
</string-array>
```

Spinners

Aby załadować tak zdefiniowane dane, należy użyć adaptera, metoda CreateFromResource pozwala stworzyć ArrayAdapter z tablicy Stringów, jeśli dane mają zostać załadowane z bazy danych należy skorzystać z CursorAdapter zamiast z ArrayAdapter.

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource( context: this,
        R.array.days, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

Spinners - obsługa eventu

Aby uzyskać dostęp do zaznaczonej przez użytkownika pozycji musimy zaimplementować klasę **`AdapterView.OnItemSelectedListener`** z poniższymi metodami

```
class CustomOnItemSelectedListener implements AdapterView.OnItemSelectedListener {


    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        System.out.println("Wybrałes " + adapterView.getItemAtPosition((int) l));
    }

    //odpalany tylko jeśli nie mamy danych w spinerze
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
    }


}
```

Następnie w metodzie spinnera przekazujemy naszą zaimplementowana klase

```
spinner.setOnItemSelectedListener(new CustomOnItemSelectedListener());
```

Toogle Button

Komponent, który pozwala na przejście pomiędzy dwoma stanami

```
<ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ToggleButton" />
```

Zdefiniowanie akcji, która ma się wykonać po zmianie stanu Toogle Button :

```
toggle = (ToggleButton) findViewById(R.id.toggleButton);
toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            System.out.println("Ustawiles On");
        } else {
            System.out.println("Ustawiles Off");
        }
    }
});
```

Dziękuję za uwagę