

Intencje w Android

Podstawą tej instrukcji jest projekt „Hello World” wykonany poprzednio.

Czy podoba ci się to co widzisz?

OCZYWIŚCIE

PRAWIE IDEALNIE

Na końcu pliku activity_main.xml dodamy kolejny Layout w którym umieścimy dodatkowe przyciski. Można to zrobić kopując już istniejący wewnętrzny LinearLayory zmieniając odpowiednio dane.

The screenshot shows the Android Studio interface with two tabs open: 'activity_main.xml' and 'MainActivity.kt'. The 'activity_main.xml' tab is active, displaying the XML code for the main activity's layout. The 'Design' tab is selected in the top right, showing a preview of the UI for a 'Pixel 3 (1080 x 2160)' screen. The UI consists of a white background with a centered text 'Czy podoba ci się to co widzisz?' and two purple buttons below it: 'OCZYWIŚCIE' and 'PRAWIE IDEALNIE'. In the XML code, there is a nested LinearLayout containing two Button elements. The first button has the text 'moja WWW' and the second has 'mój FaceBook'. The XML code is as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Czy podoba ci się to co widzisz?"/>
    <LinearLayout
        android:id="@+id/Layout3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button3"
            style="@style/Widget.AppCompat.Button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="moja WWW" />
        <Button
            android:id="@+id/button4"
            style="@style/Widget.AppCompat.Button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="mój FaceBook" />
    

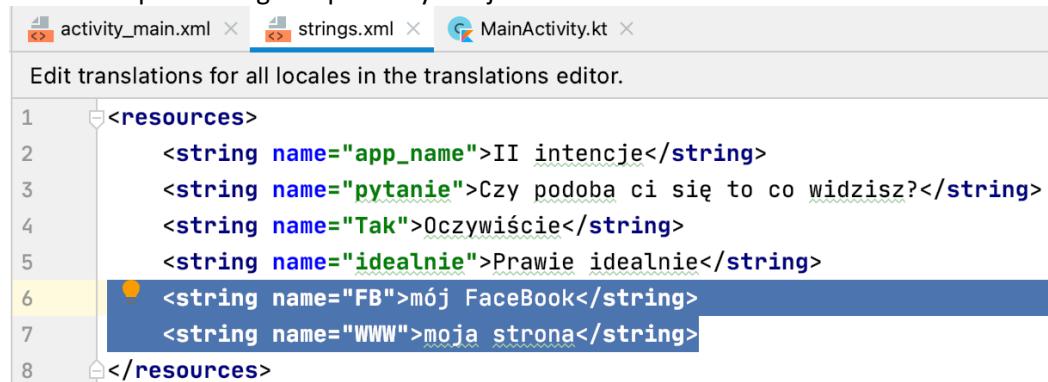

```

Kurs Android Studio

Wprowadzamy modyfikację w pliku string.xml wprowadzając nazwy przycisków

```
58             android:gravity="center"
59             android:text="moja WWW" />
60     Extract string resource
61 X Suppress: Add tools:ignore="HardcodedText" attribute
```

Zawartość pliku string.xml po modyfikacji:



```
activity_main.xml strings.xml MainActivity.kt
Edit translations for all locales in the translations editor.

1 <resources>
2     <string name="app_name">II intencje</string>
3     <string name="pytanie">Czy podoba ci się to co widzisz?</string>
4     <string name="Tak">Oczywiście</string>
5     <string name="idealnie">Prawie idealnie</string>
6     <string name="FB">mój FaceBook</string>
7     <string name="WWW">moja strona</string>
8 </resources>
```

Przechodzimy do pliku MainActivity.kt i tworzymy obsługę przycisków uruchamiające stronę WWW oraz FB:

```
33     Toast.makeText( context: this, text: "Przycisk 2 został kliknięty", Toast.LENGTH_SHORT).show()
34 }
35 val btn3_click = findViewById<Button>(R.id.button3)
36 val btn4_click = findViewById<Button>(R.id.button4)
37 btn3_click.setOnClickListener{ it: View!
38     var message3 = Toast.makeText( context: this, text: "Dziękuję za odwiedzenie mojej strony", Toast.LENGTH_SHORT)
39     message3.show()
40 }
41 btn4_click.setOnClickListener{ it: View!
42     var message4 = Toast.makeText( context: this, text: "Dołącz do moich znajomych", Toast.LENGTH_SHORT)
43     message4.show()
44 }
```

Teraz dodajemy dla każdego przycisku linki do stron jako zmienne typu uri. Jednocześnie uruchamiamy obiekt *intence* jako intencję niejawną. (intencja niejawną jest odwołaniem do intencji funkcjonującej w systemie, czy my jej nie tworzymy).

```
40     btn3_click.setOnClickListener{ it: View!
41         var message3 = Toast.makeText( context: this, text: "Dziękuję za odwiedzenie mojej strony", Toast.LENGTH_SHORT)
42         message3.show()
43         var adres = "https://bab.la/" //przykładowy słownik ang.
44         var channelRevolShen =Intent(ACTION_VIEW, Uri.parse(adres))
45     }
46     btn4_click.setOnClickListener{ it: View!
47         var message4 = Toast.makeText( context: this, text: "Dołącz do moich znajomych", Toast.LENGTH_SHORT)
48         message4.show()
49         var adres = "https://pl-pl.facebook.com/" //strona główna FB
50         var channelRevolShen =Intent(ACTION_VIEW, Uri.parse(adres))
51 }
```

Dodatkowo nasze „import ...” powinny zostać uzupełnione do postaci:

```
3 import android.annotation.SuppressLint
4 import android.content.Intent
5 import android.content.Intent.ACTION_VIEW
6 import android.net.Uri
7 import androidx.appcompat.app.AppCompatActivity
8 import android.os.Bundle
9 import android.view.View
10 import android.widget.Toast
11 import android.widget.Button
12 import android.widget.TextView
```

Teraz musimy uruchomić naszą aktywność. Dopusujemy przy **każdym** przycisku linię startową.

```
51     var channelRevolShen =Intent(ACTION_VIEW, Uri.parse(adres))
52     startActivity(channelRevolShen)
```

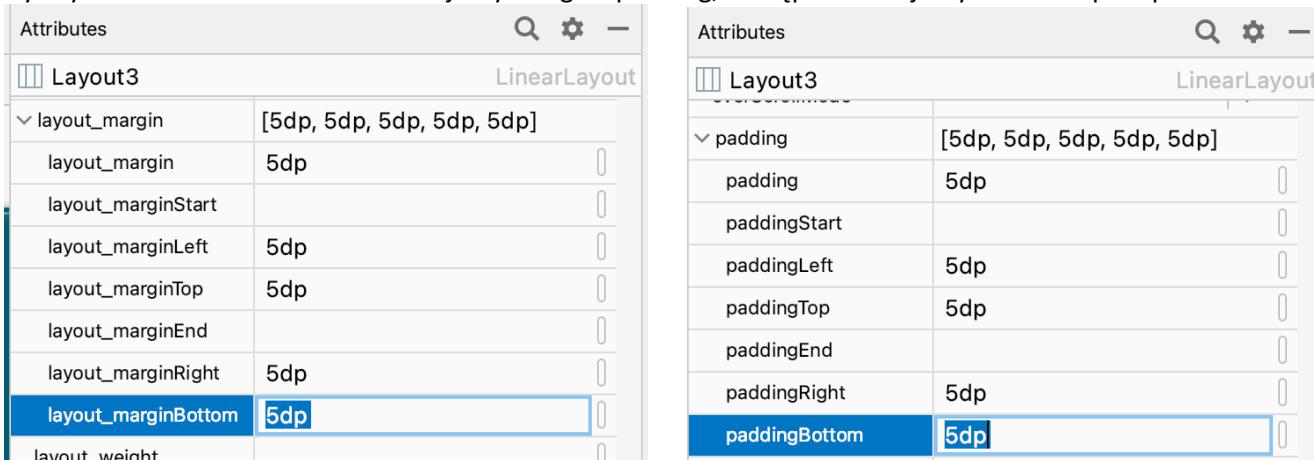
Przyciski powinny już działać

Zmieńmy wygląd aplikacji

1. Odsunięcie przycisków od siebie za pomocą paddingu i marginu
2. Zmiana koloru przycisków
3. Zmiana tła aplikacji
4. Powiększenie czcionki napisu nad przyciskami.

Ad.1

Klikamy Layout3 i w All Attributes odszukujemy margin i padding, następnie dodajemy wartość np. 5dp



W kodzie zostaną dodane atrybuty, które możemy jednocześnie skopiować np. do obu przycisków. Ja skopuję tylko padding.

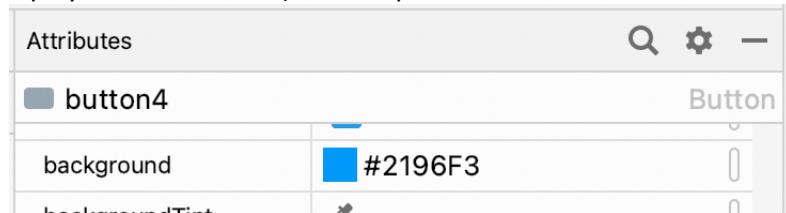
Mogemy również wyedytować kod xml podając tylko margin i padding bez rozgraniczenia kierunku: top, bottom, left lub right. Otrzymamy kod:

```
49 <LinearLayout
50     android:id="@+id/Layout3"
51     android:layout_width="wrap_content"
52     android:layout_height="wrap_content"
53     android:layout_gravity="center"
54     android:gravity="center"
55     android:orientation="horizontal"
56     android:layout_margin="5dp"
57     android:padding="5dp" >
58
59 <Button
60     android:id="@+id/button3"
61     style="@style/Widget.AppCompat.Button"
62     android:layout_width="wrap_content"
63     android:layout_height="wrap_content"
64     android:gravity="center"
65     android:text="@string/WWW"
66     android:layout_margin="8dp" />
67
68 <Button
69     android:id="@+id/button4"
70     style="@style/Widget.AppCompat.Button"
71     android:layout_width="wrap_content"
72     android:layout_height="wrap_content"
73     android:gravity="center"
74     android:text="@string/FB"
75     android:layout_margin="8dp" />
```

To samo robimy dla przycisków stosując margin np. 5dp:

Kurs Android Studio

Ad.2 Kolory przycisków zmieniamy również w atrybutach podając backgroundcolor. Wybiorę przycisk WWW jako zielony, a przycisk FB niebieski (można wpisać wartości szesnastkowe:

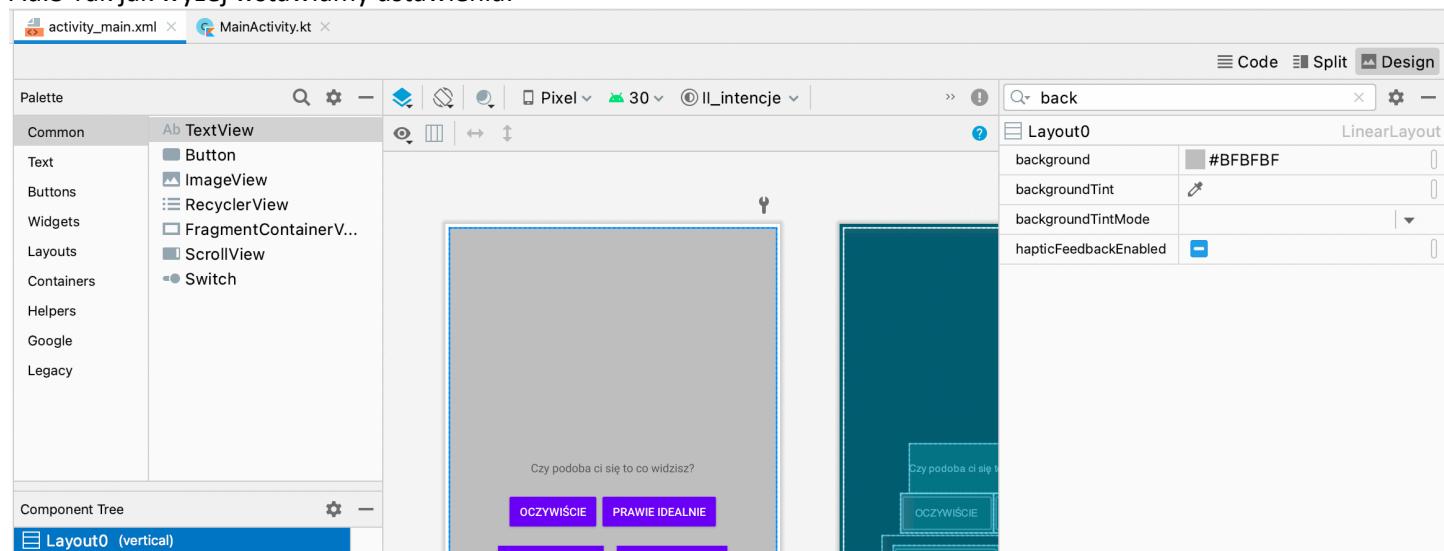


Oczywiście w kodzie zostanie dodana linia kodu definiująca kolor i pokazująca go na marginesie:

```
81 <Button  
82     android:id="@+id/button4"  
83     style="@style/Widget.AppCompat.Button"  
84     android:layout_width="wrap_content"  
85     android:layout_height="wrap_content"  
86     android:background="#2196F3" [Red arrow points here]  
87     android:gravity="center"  
88 >
```

(Uwaga: Tutaj można utworzyć własny styl. Jak utworzyć styl jest pokazane po pkt. 4.)

Ad.3 Tak jak wyżej wstawiamy ustawienia:



Mogą stworzyć również własny styl i w nim umieścić atrybuty. (własny styl zostanie utworzony po pkt. 4.)

Ad.4

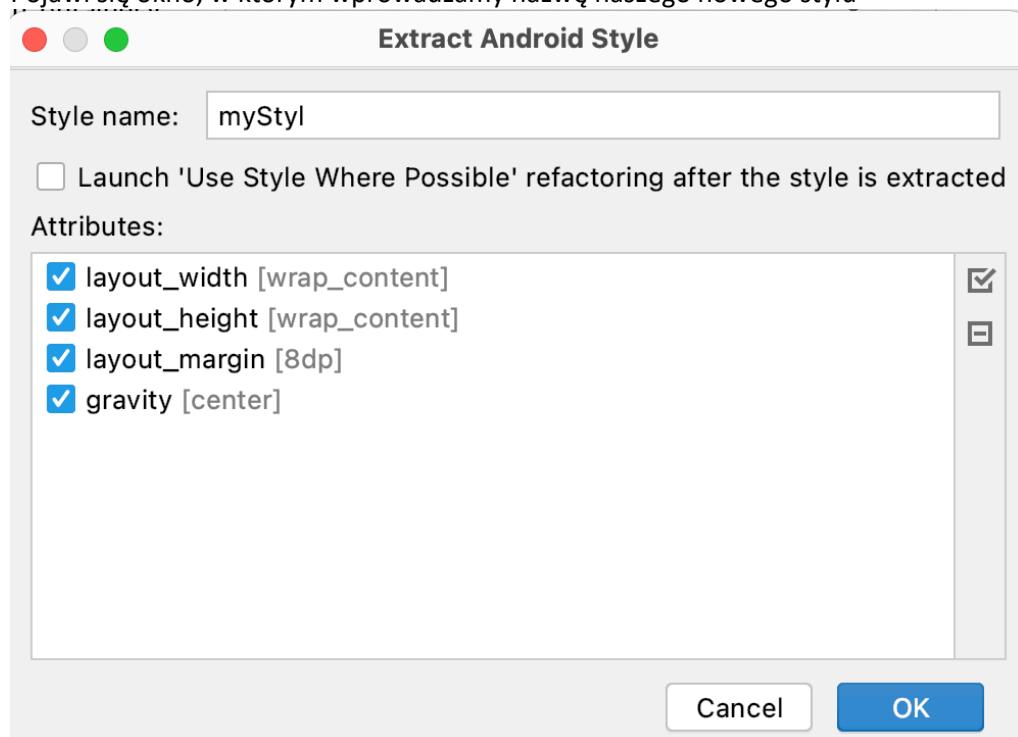
W standardowy sposób zmienimy wielkość i kolor czcionki napisu. W kodzie można dopisać dwie linie:

```
15 <TextView  
16     android:id="@+id/textView"  
17     android:layout_width="wrap_content"  
18     android:layout_height="wrap_content"  
19     android:layout_gravity="center"  
20     android:gravity="center"  
21     android:padding="20dp"  
22     android:text="Czy podoba ci się to co widzisz?"  
23     android:textColor="#FFFF00" [Red arrow points here]  
24     android:textSize="20sp" />
```

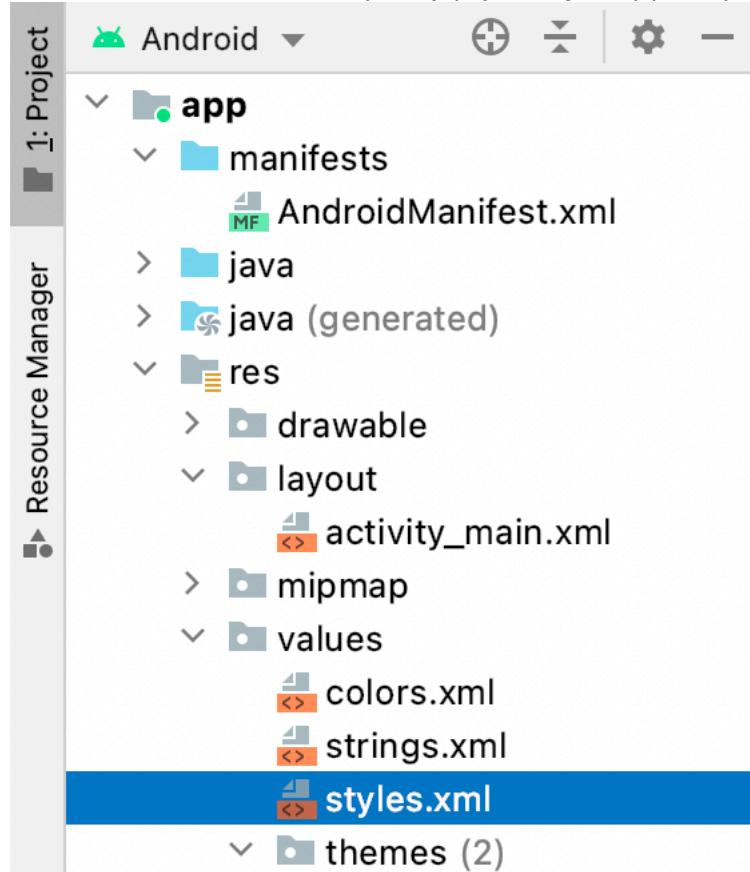
Utworzenie własnego stylu

W celu utworzenia własnego motywu/stylu należy prawym przyciskiem myszy kliknąć w przycisk i wybrać:
Refactor > Extract Style

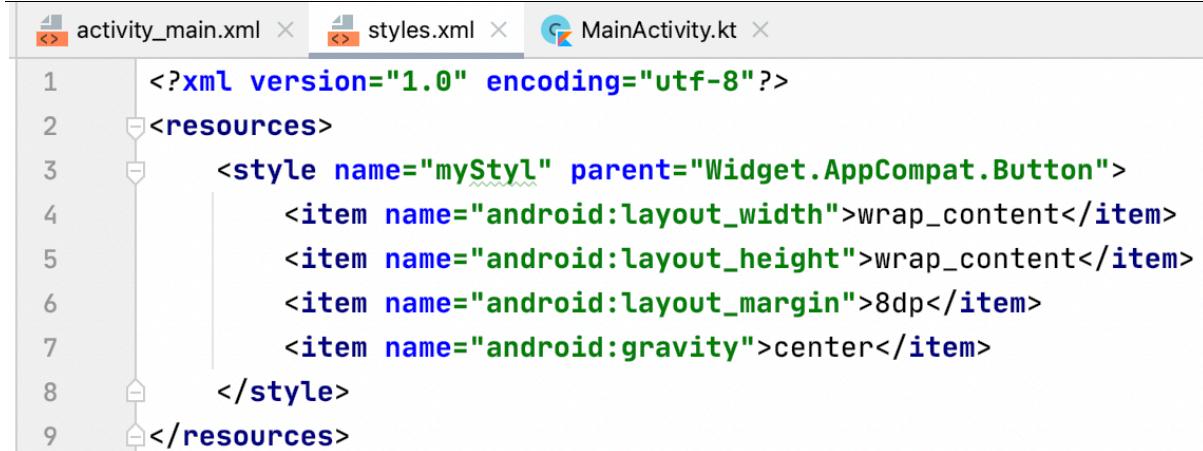
Pojawi się okno, w którym wprowadzamy nazwę naszego nowego stylu



Po zatwierdzeniu w drzewie aplikacji pojawi się nowy plik styles.xml

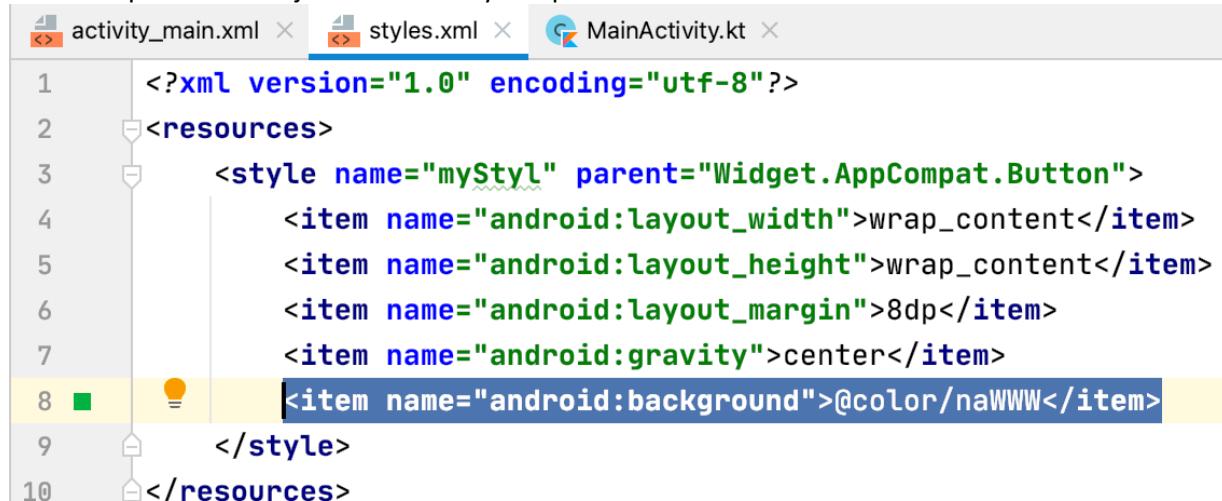


Plik **styles.xml** ma zawartość:



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="myStyl" parent="Widget.AppCompat.Button">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_margin">8dp</item>
        <item name="android:gravity">center</item>
    </style>
</resources>
```

Można dopisać linii kolejne linie kodu stylu. Np.:



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="myStyl" parent="Widget.AppCompat.Button">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_margin">8dp</item>
        <item name="android:gravity">center</item>
        <item name="android:background">@color/nawWWW</item>
    </style>
</resources>
```

Postać naszego kody przycisków możemy zmienić na:



```
<Button
    android:id="@+id/button3"
    style="@style/myStyl"
    android:text="@string/WWW" />

<Button
    android:id="@+id/button4"
    style="@style/myStyl"
    android:text="@string/FB" />
```