

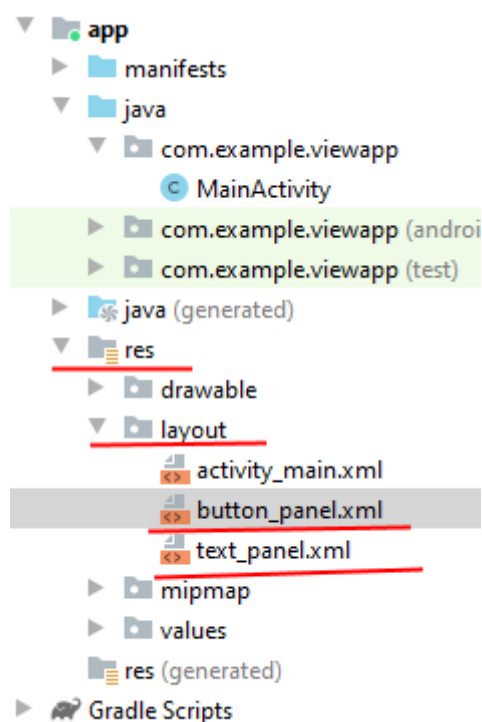
Вложенные layout

Последнее обновление:



Одна layout может содержать другую layout. Для этого применяется элемент **include**.

Например, добавим в папку **res/layout** два файла layout, которые пусть будут называться **text_panel.xml** и **button_panel.xml**:



В файле **text_panel.xml** определим следующий код:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="wrap_content"
6     android:layout_height="wrap_content">
7     <TextView
8         android:id="@+id/clicksText"
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:textSize="30sp"
12        android:text="0 Clicks"
13        app:layout_constraintLeft_toLeftOf="parent"
14        app:layout_constraintTop_toTopOf="parent"
15    />
16 </androidx.constraintlayout.widget.ConstraintLayout>
```

По сути здесь просто определено поле TextView для вывода текста.

В файле **button_panel.xml** определим следующую разметку:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="wrap_content"
6     android:layout_height="wrap_content">
7     <Button
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="Click"
11        android:onClick="onClick"
12        app:layout_constraintLeft_toLeftOf="parent"
13        app:layout_constraintTop_toTopOf="parent"
14    />
15
16 </androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определена кнопка, нажатия которой мы будем обрабатывать.

Основным файлом разметки, который определяет интерфейс приложения, по-прежнему является **activity_main.xml**. Изменим его:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
```

```

6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:padding="16dp"
9   tools:context=".MainActivity">
10
11   <include
12       android:id="@+id/textView"
13       layout="@layout/text_panel"
14       android:layout_width="wrap_content"
15       android:layout_height="wrap_content"
16       app:layout_constraintLeft_toLeftOf="parent"
17       app:layout_constraintTop_toTopOf="parent"
18       app:layout_constraintBottom_toTopOf="@+id/button"
19   />
20   <include
21       android:id="@+id/button"
22       layout="@layout/button_panel"
23       android:layout_width="wrap_content"
24       android:layout_height="wrap_content"
25       app:layout_constraintLeft_toLeftOf="parent"
26       app:layout_constraintTop_toBottomOf="@+id/textView"
27   />
28
29 </androidx.constraintlayout.widget.ConstraintLayout>

```

С помощью `ConstraintLayout` весь интерфейс здесь организуется в виде вертикального стека. С помощью элементов **include** внутри `ConstraintLayout` добавляется содержимое файлов `text_panel.xml` и `button_panel.xml`. Для указания названия файла применяется атрибут **layout**.

Это все равно, что если бы мы напрямую вместо элемента `include` добавили содержимое файлов. Однако такой способ имеет свои преимущества. Например, какая-то часть разметки, группа элементов управления может повторяться в различных `activity`. И чтобы не определять по сто раз эти элементы, можно вынести их в отдельный файл `layout` и с помощью `include` подключать их.

После добавления в `ConstraintLayout` к элементам **include** можно применять все те стандартные атрибуты, которые применяются в этом контейнере к вложенным элементам, например, настроить размеры, расположение. Также стоит отметить, что добавлять внешние `layout` можно не только в `ConstraintLayout`, но и в другие контейнеры (`LinearLayout`, `RelativeLayout` и т.д.)

Также изменим код **MainActivity**:

```

1   package com.example.viewapp;
2
3   import androidx.appcompat.app.AppCompatActivity;
4

```

```

5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.TextView;
8
9 public class MainActivity extends AppCompatActivity {
10
11     int clicks = 0;
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17
18     public void onClick(View view){
19         TextView clicksText = findViewById(R.id.clicksText);
20         clicks++;
21         clicksText.setText(clicks + " Clicks");
22     }
23 }

```

В MainActivity мы можем обращаться к элементам во вложенных файлах layout. Например, мы можем установить обработчик нажатия кнопки, в котором при нажатии изменять текст в TextView.



При этом мы несколько раз можем добавлять в один файл layout другой файл layout. Для этого вначале изменим файл **button_panel.xml** следующим образом:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="wrap_content"
6     android:layout_height="wrap_content"
7     android:background="#3F51B5"
8     android:paddingTop="10dp"
9     android:paddingBottom="10dp">
10    <Button
11        android:id="@+id/clickBtn"
12        android:layout_width="wrap_content"
13        android:layout_height="wrap_content"
14        app:layout_constraintLeft_toLeftOf="parent"
15        app:layout_constraintTop_toTopOf="parent"
16        />
17
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

И изменим файл **activity_main.xml**:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:padding="16dp"
9     tools:context=".MainActivity">
10
11    <include
12        layout="@layout/text_panel"
13        android:layout_width="wrap_content"
14        android:layout_height="wrap_content"
15        app:layout_constraintLeft_toLeftOf="parent"
16        app:layout_constraintTop_toTopOf="parent"
17        />
18    <include layout="@layout/button_panel"
19        android:id="@+id/plus_button"
20        android:layout_width="wrap_content"
21        android:layout_height="wrap_content"
22        app:layout_constraintLeft_toLeftOf="parent"
23        app:layout_constraintBottom_toBottomOf="parent"
24        app:layout_constraintRight_toLeftOf="@+id/minus_button"/>
25
```

```

26     <include layout="@layout/button_panel"
27         android:id="@+id/minus_button"
28         android:layout_width="wrap_content"
29         android:layout_height="wrap_content"
30         android:layout_marginLeft="36dp"
31         app:layout_constraintLeft_toRightOf="@id/plus_button"
32         app:layout_constraintBottom_toBottomOf="parent"/>
33
34 </androidx.constraintlayout.widget.ConstraintLayout>

```

Теперь файл button_panel.xml добавляется два раза. Важно, что при добавлении этого файла каждому элементу include присвоен определенный id. По этому id мы сможем узнать, о каком именно элементе include идет речь.

Также изменим MainActivity:

```

1  package com.example.viewapp;
2
3  import androidx.appcompat.app.AppCompatActivity;
4
5  import android.os.Bundle;
6  import android.view.View;
7  import android.widget.Button;
8  import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11
12     int clicks = 0;
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         View plusButtonView = findViewById(R.id.plus_button);
19         View minusButtonView = findViewById(R.id.minus_button);
20         TextView clicksText = findViewById(R.id.clicksText);
21
22         Button plusButton = plusButtonView.findViewById(R.id.clickBtn);
23         Button minusButton = minusButtonView.findViewById(R.id.clickBtn);
24
25         plusButton.setText("+");
26         minusButton.setText("-");
27
28         plusButton.setOnClickListener(v -> {
29             clicks++;
30             clicksText.setText(clicks + " Clicks");
31         });
32         minusButton.setOnClickListener(v -> {
33             clicks--;

```

```
34         clicksText.setText(clicks + " Clicks");
35     });
36 }
37 }
```

Здесь вначале мы получаем отдельные элементы `include` по `id`. Затем в рамках этих элементов получаем кнопку. После этого мы можем установить у кнопки любой текст и повесить обработчик события нажатия. И таким образом, поведение обеих кнопок будет различаться.



2 Clicks

