Определение интерфейса в файле XML. Файлы layout

Последнее обновление:

Как правило, для определения визуального интерфейса в проектах под Android используются специальные файлы xml. Эти файлы являются ресурсами разметки и хранят определение визуального интерфейса в виде кода XML. Подобный подход напоминает создание веб-сайтов, когда интерфейс определяется в файлах html, а логика приложения - в коде javascript.

Объявление пользовательского интерфейса в файлах XML позволяет отделить интерфейс приложения от кода. Что означает, что мы можем изменять определение интерфейса без изменения кода java. Например, в приложении могут быть определены разметки в файлах XML для различных ориентаций монитора, различных размеров устройств, различных языков и т.д. Кроме того, объявление разметки в XML позволяет легче визуализировать структуру интерфейса и облегчает отладку.

Файлы разметки графического интерфейса располагаются в проекте в каталоге **res/layout**. По умолчанию при создании проекта с пустой activity уже есть один файл ресурсов разметки **activity_main.xml**, который может выглядеть примерно так:

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

В файле определяются все графические элементы и их атрибуты, которые составляют интерфейс. При создании разметки в XML следует соблюдать некоторые правила: каждый файл разметки должен содержать один корневой элемент, который должен представлять объект **View** или **ViewGroup**.

В данном случае корневым элементом является элемент **ConstraintLayout**, который содержит элемент **TextView**.

Как правило, корневой элемент содержит определение используемых пространств имен XML. Например, в коде по умолчанию в ConstraintLayout мы можем увидеть такие атрибуты:

```
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"
```

Каждое пространство имен задается следующим образом: `xmlns:префикс="название_ресурса"`. Например, в

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

Название ресурса (или URI - Uniform Resource Indicator) - `"http://schemas.android.com/apk/res/android"`. И этот ресурс сопоставляется с префиксом `android` (`xmlns:android`). То есть через префикс мы сможем ссылаться на функциональность этого пространства имен.

Каждое пространство имен определяет некоторую функциональность, которая используется в приложении, например, предоставляют теги и атрибуты, которые необходимые для построения приложения.

- `xmlns:android="http://schemas.android.com/apk/res/android"`: содержит основные атрибуты, которые предоставляются платформой Android, применяются в

элементах управления и определяют их визуальные свойства (например, размер, позиционирование). Например, в коде ConstraintLayout используется следующий атрибут из пространства имен "http://schemas.android.com/apk/res/android":

```
android:layout_width="match_parent"
```

- `xmlns:app="http://schemas.android.com/apk/res-auto"`: содержит атрибуты, которые определены в рамках приложения. Например, в коде TextView:

```
app:layout_constraintBottom_toBottomOf="parent"
```

- `xmlns:tools="http://schemas.android.com/tools"`: применяется для работы с режиме дизайнера в Android Studio

Это наиболее распространенные пространства имен. И обычно каждый корневой элемент (не обязательно только ConstraintLayout) их содержит. Однако, если вы не планируете пользоваться графическим дизайнером в Android Studio и хотите работать целиком в коде xml, то соответственно смысла в пространстве имен "http://schemas.android.com/tools" нет, и его можно убрать.

При компиляции каждый XML-файл разметки компилируется в ресурс View. Загрузка ресурса разметки осуществляется в методе Activity.onCreate. Чтобы установить разметку для текущего объекта activity, надо в метод **setContentView()** в качестве параметра передать ссылку на ресурс разметки.

```
setContentView(R.layout.activity_main);
```

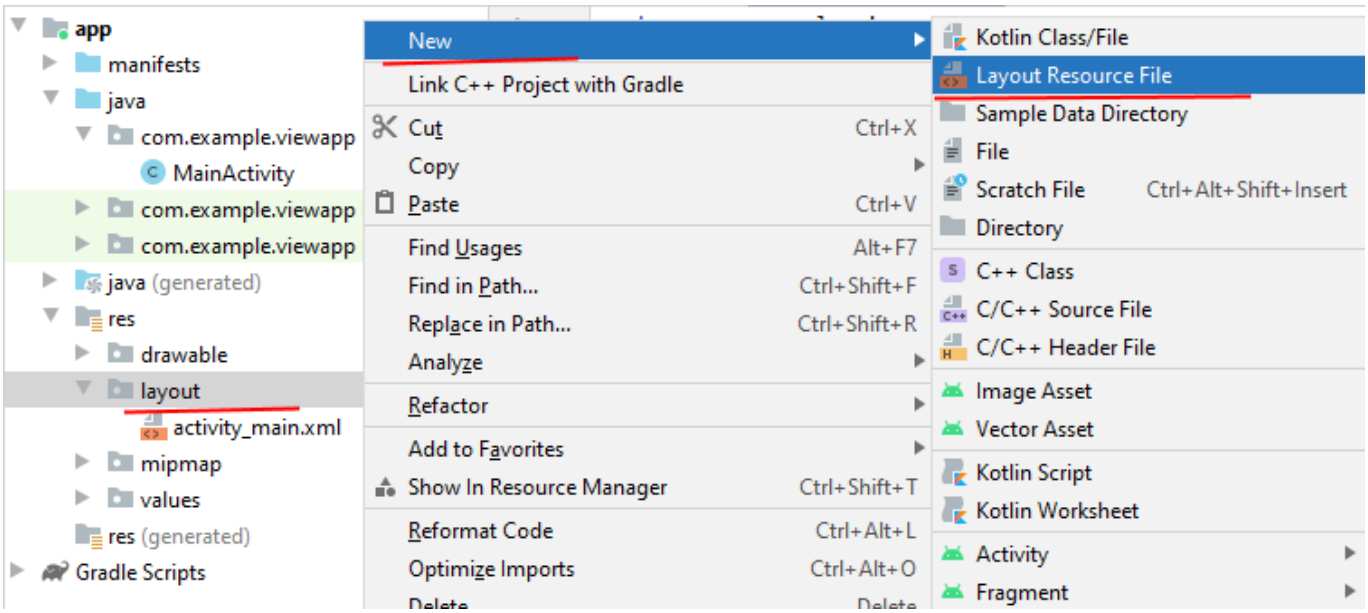
Для получения ссылки на ресурс в коде java необходимо использовать выражение `R.layout.[название_ресурса]`. Название ресурса layout будет совпадать с именем файла, поэтому чтобы использовать файл **activity_main.xml** в качестве источника визуального интерфейса, можно определить следующий код в классе **MainActivity**:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // загрузка интерфейса из файла activity_main.xml
        setContentView(R.layout.activity_main);
    }
}
```

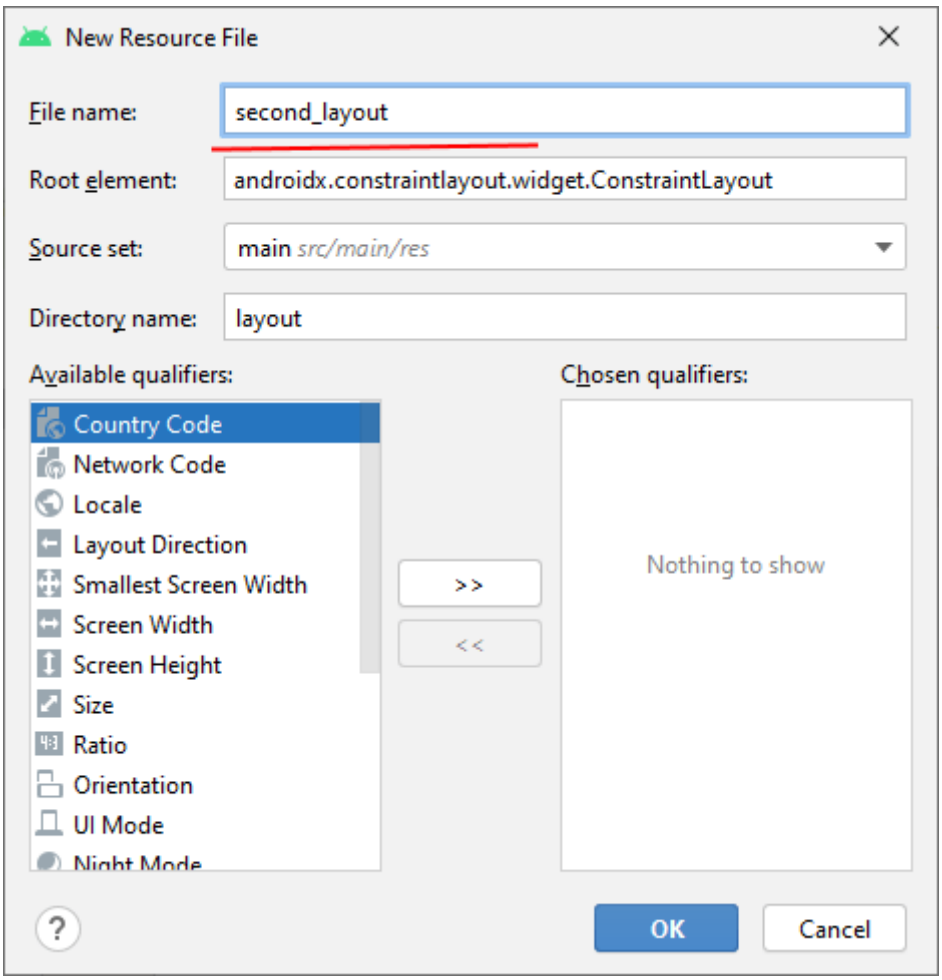
Добавление файла layout

Но у нас может быть и несколько различных ресурсов layout. Как правило, каждый отдельный класс Activity использует свой файл layout. Либо для одного класса Activity может использоваться сразу несколько различных файлов layout.

К примеру, добавим в проект новый файл разметки интерфейса. Для этого нажмем на папку **res/layout** правой кнопкой мыши и в появившемся меню выберем пункт **New -> Layout Resource File**:



После этого в специальном окошке будет предложено указать имя и корневой элемент для файла layout:



В качестве названия укажем **second_layout**. Все остальные настройки оставим по умолчанию:

- в поле **Root element** указывается корневой элемент. По умолчанию это **androidx.constraintlayout.widget.ConstraintLayout**.
- поле **Source set** указывает, куда помещать новый файл. По умолчанию это **main** - область проекта, с которой мы собственно работаем при разаботке приложения.
- поле **Directory main** указывает папку в рамках каталога, выбранного в предыдущей опции, в который собственно помещается новый файл. По умолчанию для файлов с разметкой интерфейса это **layout**.

После этого в папку **res/layout** будет добавлен новый файл **second_layout.xml**, с которым мы можем работать точно также, как и с activity_main.xml. В частности, откроем файл second_layout.xml и изменим его содержимое следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
```

```
        android:id="@+id/header"
        android:text="Welcome to Android"
        android:textSize="26sp"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определено текстовое поле TextView, которое имеет следующие атрибуты:

- `android:id` - идентификатор элемента, через который мы сможем ссылаться на него в коде. В записи `android:id="@+id/header"` символ @ указывает XML-парсеру использовать оставшуюся часть строки атрибута как идентификатор. А знак + означает, что если для элемента не определен id со значением header, то его следует определить.
- `android:text` - текст элемента - на экран будет выводиться строка "Welcome to Android".
- `android:textSize` - высота шрифта (здесь 26 единиц)
- `android:layout_width` - ширина элемента. Значение "match_parent" указывает, что элемент будет растягиваться по всей ширине контейнера ConstraintLayout
- `android:layout_height` - высота элемента. Значение "match_parent" указывает, что элемент будет растягиваться по всей высоте контейнера ConstraintLayout

Применим этот файл в качестве определения графического интерфейса в классе MainActivity:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Файл интерфейса называется second_layout.xml, поэтому по умолчанию для него будет создаваться ресурс `R.layout.second_layout`. Соответственно, чтобы его использовать, мы передаем его в метода setContentView. В итоге мы увидим на экране следующее:

Получение и управлене визуальными элементами в коде

Выше определенный элемент TextView имеет один очень важный атрибут - id или идентификатор элемента. Этот идентификатор позволяет обращаться к элементу, который определен в файле xml, из кода Java. Например, перейдем к классу MainActivity и изменим его код:

```
1  package com.example.viewapp;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class MainActivity extends AppCompatActivity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         // устанавливаем в качестве интерфейса файл second_layout.xml
13         setContentView(R.layout.second_layout);
14
15         // получаем элемент textView
16         TextView textView = findViewById(R.id.header);
17         // переустанавливаем у него текст
```

```
18          textView.setText("Hello from Java!");
19      }
20  }
```

С помощью метода `setContentView()` устанавливается разметка из файла *second_layout.xml*.

Другой важный момент, который стоит отметить - получение визуального элемента TextView. Так как в его коде мы определили атрибут `android:id`, то через этот id мы можем его получить.

Для получения элементов по id класс Activity имеет метод **findViewById()**. В этот метод передается идентификатор ресурса в виде **R.id.[идентификатор_элемента]**. Этот метод возвращает объект View - объект базового класса для всех элементов, поэтому результат метода еще необходимо привести к типу TextView.

Далее мы можем что-то сделать с этим элементом, в данном случае изменяем его текст.

Причем что важно, получение элемента происходит после того, как в методе setContentView была установлена разметка, в которой этот визуальный элемент был определен.

И если мы запустим проект, то увидим, что TextView выводит новый текст: