

TableLayout

Последнее обновление:



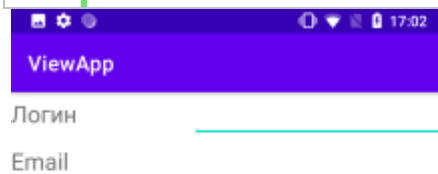
Контейнер **TableLayout** структурирует элементы управления в виде таблицы по столбцам и строкам. Определим в файле **activity_main.xml** элемент TableLayout, который будет включать две строки и два столбца:

```
1 <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:layout_width="match_parent"
3     android:layout_height="match_parent">
4     <TableRow>
5         <TextView
6             android:layout_weight="0.5"
7             android:text="Логин"
8             android:layout_width="wrap_content"
9             android:layout_height="wrap_content" />
10
11         <EditText
12             android:layout_weight="1"
13             android:layout_width="match_parent"
14             android:layout_height="wrap_content" />
15     </TableRow>
16
17     <TableRow>
18         <TextView
19             android:layout_weight="0.5"
```

```

20         android:text="Email"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content" />
23
24     <EditText
25         android:layout_weight="1"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content" />
28 </TableRow>
29 </TableLayout>

```



Используя элемент **TableRow**, мы создаем отдельную строку. Как разметка узнает сколько столбцов надо создать? Android находит строку с максимальным количеством виджетов одного уровня, и это количество будет означать количество столбцов. Например, в данном случае у нас определены две строки и в каждой по два элемента. Если бы в какой-нибудь из них было бы три виджета, то соответственно столбцов было бы также три, даже если в другой строке осталось бы два виджета.

Причем элемент **TableRow** наследуется от класса **LinearLayout**, поэтому мы можем к нему применять тот же функционал, что и к **LinearLayout**. В частности, для определения пространства для элементов в строке используется атрибут **android:layout_weight**.

Если какой-то элемент должен быть растянут на ряд столбцов, то мы можем растянуть его с помощью атрибута **layout_span**, который указывает на какое количество столбцов надо растянуть элемент:

```

1 <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:layout_width="match_parent"
3     android:layout_height="match_parent">

```

```
4 <TableRow>
5     <TextView
6         android:textSize="22sp"
7         android:text="Логин"
8         android:layout_width="100dp"
9         android:layout_height="wrap_content" />
10
11     <EditText
12         android:textSize="22sp"
13         android:layout_width="200dp"
14         android:layout_height="wrap_content" />
15 </TableRow>
16
17 <TableRow>
18     <TextView
19         android:textSize="22sp"
20         android:text="Email"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content" />
23
24     <EditText
25         android:textSize="22sp"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content" />
28 </TableRow>
29 <TableRow>
30     <Button
31         android:text="Отправить"
32         android:layout_width="wrap_content"
33         android:layout_height="wrap_content"
34         android:layout_span="2"/>
35 </TableRow>
36 </TableLayout>
```



Также можно растянуть элемент на всю строку, установив у него атрибут `android:layout_weight="1"`:

```
1 <TableRow>
2     <Button
3         android:text="Отправить"
4         android:layout_width="match_parent"
5         android:layout_height="wrap_content"
6         android:layout_weight="1" />
7 </TableRow>
```

Программное создание `TableLayout`

Создадим `TableLayout` программным образом, переложив на код java самый первый пример из данной статьи:

```
1 package com.example.viewapp;
2
3 import androidx.appcompat.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.widget.EditText;
6 import android.widget.TableLayout;
7 import android.widget.TableRow;
8 import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
```

```
14     super.onCreate(savedInstanceState);
15
16     TableLayout tableLayout = new TableLayout( this);
17
18     // первая строка
19     TableRow tableRow1 = new TableRow(this);
20
21     TextView textView1 = new TextView(this);
22     textView1.setText("Логин");
23     tableRow1.addView(textView1, new TableRow.LayoutParams(
24         TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WI
25
26     EditText editText1 = new EditText(this);
27     tableRow1.addView(editText1, new TableRow.LayoutParams(
28         TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WI
29
30     // вторая строка
31     TableRow tableRow2 = new TableRow(this);
32
33     TextView textView2 = new TextView(this);
34     textView2.setText("Email");
35     tableRow2.addView(textView2, new TableRow.LayoutParams(
36         TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WI
37
38     EditText editText2 = new EditText(this);
39     tableRow2.addView(editText2, new TableRow.LayoutParams(
40         TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WI
41
42     tableLayout.addView(tableRow1);
43     tableLayout.addView(tableRow2);
44     setContentView(tableLayout);
45 }
46 }
```