Ширина и высота элементов

Последнее обновление:

Все визуальные элеметы, которые мы используем в приложении, как правило, упорядочиваются на экране с помощью контейнеров. В Android подобными контейнерами служат такие классы как RelativeLayout, LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout. Все они по разному располагают элементы и управляют ими, но есть некоторые общие моменты при компоновке визуальных компонентов, которые мы сейчас рассмотрим.

Для организации элементов внутри контейнера используются параметры разметки. Для их задания в файле xml используются атрибуты, которые начинаются с префикса **layout_**. В частности, к таким параметрам относятся атрибуты **layout_height** и **layout_width**, которые используются для установки размеров и могут использовать одну из следующих опций:

- Растяжение по всей ширине или высоте контейнера с помощью значения **match_parent** (для всех контейнеров кроме ConstraintLayout) или **0dp** (для ConstraintLayout)
- Растяжение элемента до тех границ, которые достаточны, чтобы вместить все его содержимое с помощью значения **wrap_content**
- Точные размеры элемента, например `96 dp`

match_parent

Установка значения **match_parent** позволяет растянуть элемент по всей ширине или высоте контейнера. Стоит отметить, что данное значение применяется ко всем контейнерам, кроме ConstraintLayout. Например, рястянем элемент TextView по всей ширине и высоте контейнера LinearLayout:

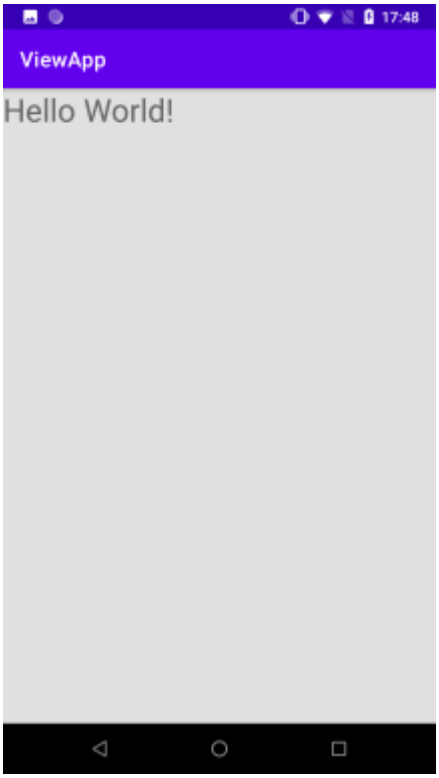
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0" />

</LinearLayout>
```

Контейнер самого верхнего уровня, в качестве которого в данном случае выступает **LinearLayout**, для высоты и ширины имеет значение **match_parent**, то есть он будет заполнять всю область для activity - как правило, весь экран.

И TextView также принимает подобные атрибуты. Значение `android:layout_width="match_parent"` обеспечивает растяжение по ширине, а `android:layout_height="match_parent"` - по вертикали. Для наглядности в TextView применяет атрибут **android:background**, который представляет фон и в данном случае окрашивает элемент в цвет "#e0e0e0", благодаря чему мы можем увидеть занимаемую им область.



Следует учитывать, что значение **match_parent** можно применять почти во всех встроенных контейнерах, типа LinearLayout или RelativeLayout и их элементах. Однако **match_parent** не рекомендуется применять к элементам внутри ConstraintLayout. Вместо "match_parent" в ConstraintLayout можно использовать значение **0dp**, чтобы растянуть элемент по горизонтали или вертикали:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />

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

Стоит отметить, что ConstraintLayout сама также растягивается по ширине и высоте экрана с помощью значения "match_parent" в атрибутах `layout_width` и `android:layout_height`, но к вложенным элементам это значение не рекомендуется применять.

Поскольку ConstraintLayout имеет некоторые особенности при установке размеров, то более подробно работа с размерами элементов именно в ConstraintLayout раскрыта более подробно в одной из следующих тем.

wrap_content

Значение wrap_content устанавливает те значения для ширины или высоты, которые необходимы, чтобы разместить на экране содержимое элемента:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    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!"
        android:textSize="30sp"
        android:background="#ffcdd2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />

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

Здесь элемент TextView растягивается до тех значений, которые достаточны для размещения его текста.

Установка точных значений

Также мы можем установить точные значения:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_height="90dp"
        android:layout_width="150dp"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

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

Кроме того, можно комбинировать несколько значений, например, растянуть по ширине содержимого и установить точные значения для высоты:

```
<TextView
        android:layout_height="80dp"
        android:layout_width="wrap_content"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
/>
```

Если для установки ширины и длины используется значение **wrap_content**, то мы можем дополнительно ограничить минимальные и максимальные значения с помощью атрибутов **minWidth/maxWidth** и **minHeight/maxHeight**:

```
<TextView
        android:minWidth="200dp"
        android:maxWidth="250dp"
        android:minHeight="100dp"
        android:maxHeight="200dp"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
```

```
/>
```

В этом случае ширина TextView будет такой, которая достаточна для вмещения текста, но не больше значения maxWidth и не меньше значения minWidth. То же самое для установки высоты.

Программная установка ширины и высоты

Если элемент, к примеру, тот же TextView создается в коде java, то для установки высоты и ширины можно использовать метод **setLayoutParams()**. Так, изменим код MainActivity:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        TextView textView = new TextView(this);
        textView.setText("Hello Android");
        textView.setTextSize(26);

        // устанавливаем параметры размеров и расположение элемента
        ConstraintLayout.LayoutParams layoutParams = new ConstraintLayout.La
                (ConstraintLayout.LayoutParams.WRAP_CONTENT, ConstraintLayou
        // эквивалент app:layout_constraintLeft_toLeftOf="parent"
        layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        // эквивалент app:layout_constraintTop_toTopOf="parent"
        layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
        // устанавливаем параметры для textView
        textView.setLayoutParams(layoutParams);
        // добавляем TextView в ConstraintLayout
        constraintLayout.addView(textView);
        setContentView(constraintLayout);
    }
}
```

В метод `setLayoutParams()` передается объект **ViewGroup.LayoutParams**. Этот объект инициализируется двумя параметрами: шириной и высотой. Для указания ширины и высоты можно использовать одну из констант

ViewGroup.LayoutParams.WRAP_CONTENT или **ViewGroup.LayoutParams.MATCH_PARENT** (в случае с LinearLayout или RelativeLayout).

Поскольку в данном случае мы имеем дело с контейнером ConstraintLayout, то для установки размеров применяется значение **ConstraintLayout.LayoutParams.WRAP_CONTENT**. В реальности класс `androidx.constraintlayout.widget.ConstraintLayout.LayoutParams`, который предоставляет это значение, наследуется от `android.view.ViewGroup.LayoutParams`

Также мы можем передать точные значения или комбинировать типы значений:

```
1  ConstraintLayout.LayoutParams layoutParams = new ConstraintLayout.LayoutPara
2                  (ConstraintLayout.LayoutParams.WRAP_CONTENT, 200);
```