1. Learn how to create new layout and how to set it into setContentview. When you turn android it shows up bit wrong to solve this problem we have to create new folder with orientation and choose from orientation landscape and there create new content with views.Then it will shows up well in landscape view and linear view

Расположение View-элементов на экране зависит от **ViewGroup**(Layout), в которой они находятся. В этом уроке мы рассмотрим основные виды **Layout**.

**1 LinearLayout**– отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical). Я использовал это на прошлом уроке, когда демонстрировал использование layout-файлов при смене ориентации.

1. **TableLayout**– отображает элементы в виде таблицы, по строкам и столбцам.
2. **RelativeLayout**– для каждого элемента настраивается его положение относительно других элементов.
3. **AbsoluteLayout**– для каждого элемента указывается явная позиция на экране в системе координат (x,y)

- слово **android** в названии каждого атрибута – это [namespace](http://www.w3schools.com/xml/xml_namespaces.asp" \t "_blank), я его буду опускать при объяснениях.  
**- id** – это ID элемента,  
**- layout\_width** (ширина элемента) и **layout\_height** (высота элемента) могут задаваться в абсолютных значениях, а могут быть следующими: **fill\_parent** (максимально возможная ширина или высота в пределах родителя) и **wrap\_content** (ширина или высота определяется по содержимому элемента). В [хелпе](http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html" \t "_blank) указывается, что есть еще **match\_parent**. Это тоже самое, что и fill\_parent. По каким-то причинам, разработчики системы решили, что название match\_parent удобнее, и от fill\_parent постепенно будут отказываться. А пока его оставили для совместимости. Так что запомните, что **match\_parent = fill\_parent** и в дальнейшем будем стараться использовать **match\_parent**. Позже мы еще остановимся на этом и разберем подробнее.

Code in MainActivity

EditText text = findViewById(R.id.***editText2***);  
TextView show = findViewById(R.id.***textView***);  
Button btn = findViewById(R.id.***button***);  
Button bt = findViewById(R.id.***button5***);  
btn.setOnClickListener(b ->{  
 show.setText(text.getText());  
});  
bt.setOnClickListener(view -> {  
 text.setText(**""**);  
 show.setText(**""**);  
});

Test2.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**LinearLayout  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:inputType="textPersonName"  
 android:text="@string/textName"  
 android:ems="10"  
 tools:layout\_editor\_absoluteY="16dp" tools:layout\_editor\_absoluteX="65dp" android:id="@+id/editText2"**/>  
 <**Button  
 android:text="@string/save"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" tools:layout\_editor\_absoluteY="105dp"  
 tools:layout\_editor\_absoluteX="38dp" android:id="@+id/button"**/>  
 <**Button  
 android:text="@string/cancel"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" tools:layout\_editor\_absoluteY="105dp"  
 tools:layout\_editor\_absoluteX="209dp" android:id="@+id/button5"**/>  
 <**TextView  
 android:text="TextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="47dp" android:id="@+id/textView"**/>  
 </**LinearLayout**>  
</**android.support.constraint.ConstraintLayout**>

**OnlickListener**

public class MainActivity extends Activity implements OnClickListener {

   TextView tvOut;

   Button btnOk;

   Button btnCancel;

   /\*\* Called when the activity is first created. \*/

   @Override

   public void onCreate(Bundle savedInstanceState) {

     super.onCreate(savedInstanceState);

     setContentView(R.layout.main);

     // найдем View-элементы

     tvOut = (TextView) findViewById(R.id.tvOut);

     btnOk = (Button) findViewById(R.id.btnOk);

     btnCancel = (Button) findViewById(R.id.btnCancel);

     // присваиваем обработчик кнопкам

     btnOk.setOnClickListener(this);

     btnCancel.setOnClickListener(this);

   }

   @Override

   public void onClick(View v) {

     // по id определеяем кнопку, вызвавшую этот обработчик

     switch (v.getId()) {

     case R.id.btnOk:

       // кнопка ОК

       tvOut.setText("Нажата кнопка ОК");

       break;

     case R.id.btnCancel:

       // кнопка Cancel

       tvOut.setText("Нажата кнопка Cancel");

       break;

     }

   }

 }

**28-09**

Для первого TextView мы сделаем контекстное меню, с помощью которого будем менять цвет текста. Для второго – будем менять размер текста.

Принцип создания контекстного меню похож на создание обычного меню. Но есть и отличия.

Метод создания [onCreateContextMenu](http://developer.android.com/reference/android/app/Activity.html" \l "onCreateContextMenu(android.view.ContextMenu,%20android.view.View,%20android.view.ContextMenu.ContextMenuInfo)" \t "_blank) вызывается каждый раз перед показом меню. На вход ему передается:

- **ContextMenu**, в который мы будем добавлять пункты  
- **View**- элемент экрана, для которого вызвано контекстное меню  
- **ContextMenu.ContextMenuInfo**– содержит доп.информацию, когда контекстное меню вызвано для элемента списка. Пока мы это не используем, но, когда будем изучать списки, увидим, что штука полезная.

Метод обработки [onContextItemSelected](http://developer.android.com/reference/android/app/Activity.html" \l "onContextItemSelected(android.view.MenuItem)" \t "_blank) аналогичный методу **onOptionsItemSelected**для обычного меню. На вход передается **MenuItem**– пункт меню, который был нажат.

Также нам понадобится третий метод [registerForContextMenu](http://developer.android.com/reference/android/app/Activity.html" \l "registerForContextMenu(android.view.View)" \t "_blank). На вход ему передается **View**и это означает, что для этой View необходимо создавать контекстное меню. Если не выполнить этот метод, контекстное меню для View создаваться не будет.

Давайте кодить, открываем **MainActivity.java**. Опишем и найдем **TextView**и укажем, что необходимо создавать для них контекстное меню.

TextView tvColor, tvSize;

  /\*\* Called when the activity is first created. \*/

  @Override

  public void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.main);

      tvColor = (TextView) findViewById(R.id.tvColor);

      tvSize = (TextView) findViewById(R.id.tvSize);

      // для tvColor и tvSize необходимо создавать контекстное меню

      registerForContextMenu(tvColor);

      registerForContextMenu(tvSize);

  }

final int MENU\_COLOR\_RED = 1;

final int MENU\_COLOR\_GREEN = 2;

final int MENU\_COLOR\_BLUE = 3;

final int MENU\_SIZE\_22 = 4;

final int MENU\_SIZE\_26 = 5;

final int MENU\_SIZE\_30 = 6;

@Override

public void onCreateContextMenu(ContextMenu menu, View v,

    ContextMenuInfo menuInfo) {

  // TODO Auto-generated method stub

  switch (v.getId()) {

case R.id.tvColor:

  menu.add(0, MENU\_COLOR\_RED, 0, "Red");

  menu.add(0, MENU\_COLOR\_GREEN, 0, "Green");

  menu.add(0, MENU\_COLOR\_BLUE, 0, "Blue");

  break;

case R.id.tvSize:

  menu.add(0, MENU\_SIZE\_22, 0, "22");

  menu.add(0, MENU\_SIZE\_26, 0, "26");

  menu.add(0, MENU\_SIZE\_30, 0, "30");

  break;

}

}

@Override

public boolean onContextItemSelected(MenuItem item) {

  // TODO Auto-generated method stub

  switch (item.getItemId()) {

  // пункты меню для tvColor

case MENU\_COLOR\_RED:

  tvColor.setTextColor(Color.RED);

  tvColor.setText("Text color = red");

  break;

case MENU\_COLOR\_GREEN:

  tvColor.setTextColor(Color.GREEN);

  tvColor.setText("Text color = green");

  break;

case MENU\_COLOR\_BLUE:

  tvColor.setTextColor(Color.BLUE);

  tvColor.setText("Text color = blue");

  break;

// пункты меню для tvSize

case MENU\_SIZE\_22:

  tvSize.setTextSize(22);

  tvSize.setText("Text size = 22");

  break;

case MENU\_SIZE\_26:

  tvSize.setTextSize(26);

  tvSize.setText("Text size = 26");

  break;

case MENU\_SIZE\_30:

  tvSize.setTextSize(30);

  tvSize.setText("Text size = 30");

  break;

  }

  return super.onContextItemSelected(item);

}

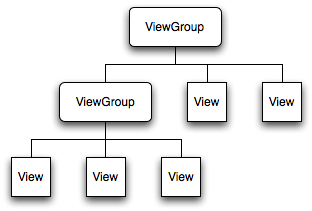
Открываем **MainActivity.java** и обратим внимание на строку:

|  |  |
| --- | --- |
| 1 | setContentView(R.layout.main); |

Напомню, что в этой строке мы указываем, что **Activity**в качестве экрана будет использовать **layout-файл main.xml**. Есть другая реализация этого метода, которая на вход принимает не layout-файл, а **View**-элемент и делает его корневым. В layout-файлах корневой элемент обычно **LinearLayout**, мы тоже используем его.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Override  public void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      // создание LinearLayout      LinearLayout linLayout = new LinearLayout(this);      // установим вертикальную ориентацию      linLayout.setOrientation(LinearLayout.VERTICAL);      // создаем LayoutParams      LayoutParams linLayoutParam = new LayoutParams(LayoutParams.MATCH\_PARENT, LayoutParams.MATCH\_PARENT);      // устанавливаем linLayout как корневой элемент экрана      setContentView(linLayout, linLayoutParam);  } |

Обновим импорт – **CTRL+SHIFT+O**. Eclipse предложит нам выбрать, какой именно **LayoutParams**мы будем использовать. Тут надо остановиться подробнее. Вспомним теорию про экраны. Экран [состоит](http://developer.android.com/guide/topics/ui/index.html#ViewHierarchy) из ViewGroup и вложенных в них View.



Известные нам примеры **ViewGroup**– это **LinearLayout**, **TableLayout**, **RelativeLayout**и т.д. Каждая из этих ViewGroup [имеет вложенный класс LayoutParams](http://developer.android.com/guide/topics/ui/declaring-layout.html#layout-params). Базовым для этих LayoutParams является [ViewGroup.LayoutParams](http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html" \t "_blank).

**ViewGroup**.**LayoutParams**имеет всего два атрибута: **height**и **width**. Его подкласс [ViewGroup.MarginLayoutParams](http://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html" \t "_blank) наследует два этих атрибута и имеет свои четыре: **bottomMargin**, **leftMargin**, **rightMargin**, **topMargin**. Класс [LinearLayout.LayoutParams](http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html" \t "_blank) в свою очередь является подклассом ViewGroup.MarginLayoutParams, наследует от него уже 6 аттрибутов и добавляет свои два: **gravity**и **weight**.

Т.е. объект **LinearLayout**имеет вложенный класс **LinearLayout.LayoutParams** с layout-аттрибутами. И эти аттрибуты распространяются на все дочерние View и ViewGroup.

**Полный Код Урока**

public class MainActi

vity extends Activity {

    /\*\* Called when the activity is first created. \*/

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        // создание LinearLayout

        LinearLayout linLayout = new LinearLayout(this);

        // установим вертикальную ориентацию

        linLayout.setOrientation(LinearLayout.VERTICAL);

        // создаем LayoutParams

        LayoutParams linLayoutParam = new LayoutParams(LayoutParams.MATCH\_PARENT, LayoutParams.MATCH\_PARENT);

        // устанавливаем linLayout как корневой элемент экрана

        setContentView(linLayout, linLayoutParam);

        LayoutParams lpView = new LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

        TextView tv = new TextView(this);

        tv.setText("TextView");

        tv.setLayoutParams(lpView);

        linLayout.addView(tv);

        Button btn = new Button(this);

        btn.setText("Button");

        linLayout.addView(btn, lpView);

        LinearLayout.LayoutParams leftMarginParams = new LinearLayout.LayoutParams(

                LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

        leftMarginParams.leftMargin = 50;

        Button btn1 = new Button(this);

        btn1.setText("Button1");

        linLayout.addView(btn1, leftMarginParams);

        LinearLayout.LayoutParams rightGravityParams = new LinearLayout.LayoutParams(

                LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

        rightGravityParams.gravity = Gravity.RIGHT;

        Button btn2 = new Button(this);

        btn2.setText("Button2");

        linLayout.addView(btn2, rightGravityParams);

    }

}

