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), 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) 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#onCreateContextMenu(android.view.ContextMenu,%20android.view.View,%20android.view.ContextMenu.ContextMenuInfo)) 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#onContextItemSelected(android.view.MenuItem)) 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#registerForContextMenu(android.view.View)). 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);

}

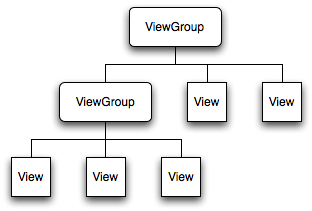
Открываем **MainActivity.java** и обратим внимание на строку:

|  |  |
| --- | --- |
| 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).

**ViewGroup**.**LayoutParams**имеет всего два атрибута: **height**и **width**. Его подкласс [ViewGroup.MarginLayoutParams](http://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html) 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) 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;

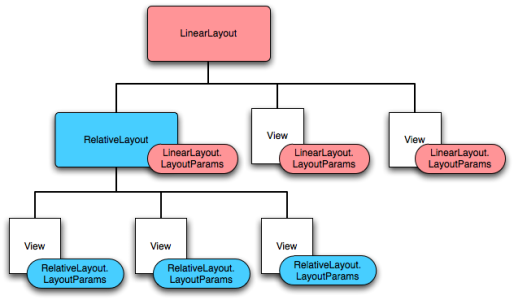
        Button btn2 = new Button(this);

        btn2.setText("Button2");

        linLayout.addView(btn2, rightGravityParams);

    }

}



рансформации конфигурируются в XML файлах, затем в коде программы считываются и присваиваются View-элементам. Я не буду дублировать [хелп](http://developer.android.com/guide/topics/resources/animation-resource.html#View) и все расписывать, а сразу перейду к практике.

В нашем проекте есть папка **res**. Надо в ней создать папку **anim**. Сделать это можно, например, так:  правой кнопкой на **res**и в меню выбираем **New**-> **Folder**. В папке **anim**надо создать файлы. Делается это аналогично: правой кнопкой на **anim**и в меню выбираем **New**-> **File**. В этих файлах будем конфигурировать анимацию.

Создаем следующие файлы в папке res/anim:

**Имя файла**: myalpha.xml

**Содержимое**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?xml version="1.0" encoding="utf-8"?>  <alpha      xmlns:android="http://schemas.android.com/apk/res/android"      android:fromAlpha="0.0"      android:toAlpha="1.0"      android:duration="3000">  </alpha> |

**Описание трансформации**: меняется прозрачность с 0 до 1 в течение трех секунд.

**Имя файла**: myscale.xml

**Содержимое**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?xml version="1.0" encoding="utf-8"?>  <scale      xmlns:android="http://schemas.android.com/apk/res/android"      android:fromXScale="0.1"      android:toXScale="1.0"      android:fromYScale="0.1"      android:toYScale="1.0"      android:pivotX="50%"      android:pivotY="50%"      android:duration="3000">  </scale> |

**Описание трансформации**: изменение размера с 0.1 от оригинальной ширины и высоты до 1. Точка, относительно которой будет производиться масштабирование, лежит ровно посередине объекта (pivotX, pivotY). Продолжительность – 3 сек.

**Имя файла**: mytrans.xml

**Содержимое**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?xml version="1.0" encoding="utf-8"?>  <translate      xmlns:android="http://schemas.android.com/apk/res/android"      android:fromXDelta="-150"      android:toXDelta="0"      android:fromYDelta="-200"      android:toYDelta="0"      android:duration="3000">  </translate> |

**Описание трансформации**: перемещение с -150 относительно текущей позиции по оси X и -200 по оси Y в текущую позицию (0,0). Продолжительность – 3 сек.

**Имя файла**: myrotate.xml

**Содержимое**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?xml version="1.0" encoding="utf-8"?>  <rotate      xmlns:android="http://schemas.android.com/apk/res/android"      android:fromDegrees="0"      android:toDegrees="360"      android:duration="3000">  </rotate> |

**Описание трансформации**: поворот относительно левого верхнего угла (т.к. не указаны pivotX, pivotY) на 360 градусов в течение трех секунд

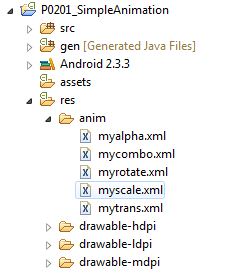
**Имя файла**: mycombo.xml

**Содержимое**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <?xml version="1.0" encoding="utf-8"?>  <set      xmlns:android="http://schemas.android.com/apk/res/android">      <rotate          android:fromDegrees="0"          android:toDegrees="360"          android:duration="3000"          android:pivotX="50%"          android:pivotY="50%">      </rotate>      <scale          android:fromXScale="0.1"          android:toXScale="1.0"          android:fromYScale="0.1"          android:toYScale="1.0"          android:pivotX="50%"          android:pivotY="50%"          android:duration="3000">      </scale>  </set> |

**Описание трансформации**: одновременно увеличение размера и вращение в течение трех секунд. Обратите внимание, для комбинации трансформ использован тег <set>

Итак, мы создали 5 файлов анимации.



И теперь можем применять их к View-компонентам.

Открываем **main.xml** и создадим экран:

package ru.startandroid.develop.p0201simpleanimation;

import android.app.Activity;

import android.os.Bundle;

import android.view.ContextMenu;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.MenuItem;

import android.view.View;

import android.view.animation.Animation;

import android.view.animation.AnimationUtils;

import android.widget.TextView;

public class MainActivity extends Activity {

  // константы для ID пунктов меню

  final int MENU\_ALPHA\_ID = 1;

  final int MENU\_SCALE\_ID = 2;

  final int MENU\_TRANSLATE\_ID = 3;

  final int MENU\_ROTATE\_ID = 4;

  final int MENU\_COMBO\_ID = 5;

  TextView tv;

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

  @Override

  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

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

    // регистрируем контекстное меню для компонента tv

    registerForContextMenu(tv);

  }

  @Override

  public void onCreateContextMenu(ContextMenu menu, View v,

      ContextMenuInfo menuInfo) {

    // TODO Auto-generated method stub

    switch (v.getId()) {

    case R.id.tv:

      // добавляем пункты

      menu.add(0, MENU\_ALPHA\_ID, 0, "alpha");

      menu.add(0, MENU\_SCALE\_ID, 0, "scale");

      menu.add(0, MENU\_TRANSLATE\_ID, 0, "translate");

      menu.add(0, MENU\_ROTATE\_ID, 0, "rotate");

      menu.add(0, MENU\_COMBO\_ID, 0, "combo");

      break;

    }

    super.onCreateContextMenu(menu, v, menuInfo);

  }

  @Override

  public boolean onContextItemSelected(MenuItem item) {

    Animation anim = null;

    // определяем какой пункт был нажат

    switch (item.getItemId()) {

    case MENU\_ALPHA\_ID:

      // создаем объект анимации из файла anim/myalpha

      anim = AnimationUtils.loadAnimation(this, R.anim.myalpha);

      break;

    case MENU\_SCALE\_ID:

      anim = AnimationUtils.loadAnimation(this, R.anim.myscale);

      break;

    case MENU\_TRANSLATE\_ID:

      anim = AnimationUtils.loadAnimation(this, R.anim.mytrans);

      break;

    case MENU\_ROTATE\_ID:

      anim = AnimationUtils.loadAnimation(this, R.anim.myrotate);

      break;

    case MENU\_COMBO\_ID:

      anim = AnimationUtils.loadAnimation(this, R.anim.mycombo);

      break;

    }

    // запускаем анимацию для компонента tv

    tv.startAnimation(anim);

    return super.onContextItemSelected(item);

  }

}

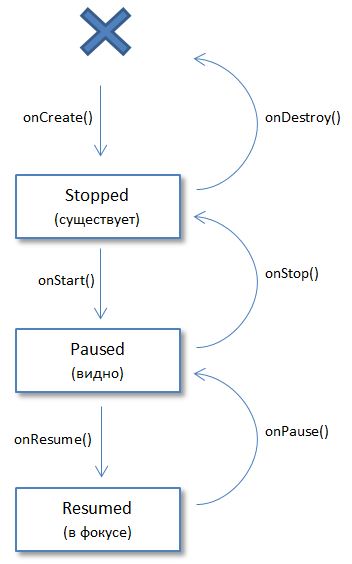
### Теория

При работе приложения, мы **создаем**новые **Activity**и **закрываем**старые, **сворачиваем**приложение, снова **открываем**и т.д. Activity умеет обрабатывать все эти движения. Это необходимо, например, для освобождения ресурсов или сохранения данных. В [хелпе](http://developer.android.com/guide/topics/fundamentals/activities.html#Lifecycle) достаточно подробно это описано.

Созданное при работе приложения **Activity**может быть в одном из **трех состояний**:

**Resumed**- Activity видно на экране, оно находится в фокусе, пользователь может с ним взаимодействовать. Это состояние также иногда называют Running.  
**Paused**- Activity не в фокусе, пользователь не может с ним взаимодействовать, но его видно (оно перекрыто другим Activity, которое занимает не весь экран или полупрозрачно).  
**Stopped**- Activity не видно (полностью перекрывается другим Activity), соответственно оно не в фокусе и пользователь не может с ним взаимодействовать.

Когда Activity переходит из одного **состояния**в другое, система вызывает различные его **методы**, которые мы можем заполнять своим кодом. Схематично это можно изобразить так:



 Для упрощения понимания я дал краткое описание состояний в скобках под названиями. А крестом обозначил отсутствие Activity.

Итак, мы имеем следующие методы Activity, которые вызывает система:

**onCreate**() – вызывается при первом создании Activity  
**onStart**() – вызывается перед тем, как Activity будет видно пользователю  
**onResume**() – вызывается перед тем как будет доступно для активности пользователя (взаимодействие)

**onPause**() – вызывается перед тем, как будет показано другое Activity  
**onStop**() – вызывается когда Activity становится не видно пользователю  
**onDestroy**() – вызывается перед тем, как Activity будет уничтожено

Т.е. эти методы **НЕ** вызывают смену состояния. Наоборот, смена состояния Activity является триггером, который вызывает эти методы. Тем самым нас уведомляют о смене, и мы можем реагировать соответственно. Посмотрим на практике, когда и в каком порядке вызываются эти методы.

### 

Передать данные через Intent

В **EditText** будем вводить имя и фамилию, а кнопка **Submit**будет вызывать другой экран и передавать ему эти данные.

Пишем код для **MainActivity.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package ru.startandroid.develop.p0281intentextras;    import android.app.Activity;  import android.content.Intent;  import android.os.Bundle;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  import android.widget.EditText;    public class MainActivity extends Activity implements OnClickListener {      EditText etFName;    EditText etLName;      Button btnSubmit;          /\*\* Called when the activity is first created. \*/      @Override      public void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.main);            etFName = (EditText) findViewById(R.id.etFName);          etLName = (EditText) findViewById(R.id.etLName);            btnSubmit = (Button) findViewById(R.id.btnSubmit);          btnSubmit.setOnClickListener(this);        }        @Override    public void onClick(View v) {      Intent intent = new Intent(this, ViewActivity.class);      intent.putExtra("fname", etFName.getText().toString());      intent.putExtra("lname", etLName.getText().toString());      startActivity(intent);    }  } |

Определяем поля ввода и кнопку. Кнопке присваиваем обработчик – Activity (this). Рассмотрим реализацию метода **onClick**.  Мы **создаем Intent**с использованием класса, а не action. Если помните, с такого способа мы начинали знакомство с Intent. Напомню - это означает, что система просмотрит манифест файл нашего приложения, и если найдет Activity с таким классом – отобразит его. ViewActivity пока не создан, поэтому код будет подчеркнут красным. Это не мешает нам сохранить файл. Чуть позже мы создадим это Activity и ошибка исчезнет.

Итак, Intent создан, смотрим код дальше. Используется метод [putExtra](http://developer.android.com/reference/android/content/Intent.html#putExtra(java.lang.String,%20java.lang.CharSequence)). Он имеет множество вариаций и аналогичен методу **put**для **Map**, т.е. **добавляет**к объекту **пару**. Первый параметр – это **ключ**(имя), второй - **значение**.

Мы поместили в Intent два объекта с именами: *fname*и *lname*. *fname* содержит значение поля etFName, *lname*– значение поля etLName. Остается только **отправить**укомплектованный **Intent**с помощью метода **startActivity**.

Теперь создадим второе Activity. Назовем его **ViewActivity**.

Создаем для него layout-файл **view.xml**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?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"      android:orientation="vertical">      <TextView          android:id="@+id/tvView"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_gravity="center\_horizontal"          android:layout\_marginTop="20dp"          android:text="TextView"          android:textSize="20sp">      </TextView>  </LinearLayout> |

Здесь просто TextView, который будет отображать пришедшие данные.

Создаем класс **ViewActivity**. И пишем код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package ru.startandroid.develop.p0281intentextras;    import android.app.Activity;  import android.content.Intent;  import android.os.Bundle;  import android.widget.TextView;    public class ViewActivity extends Activity {      TextView tvView;      @Override    protected void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.view);        tvView = (TextView) findViewById(R.id.tvView);        Intent intent = getIntent();        String fName = intent.getStringExtra("fname");      String lName = intent.getStringExtra("lname");        tvView.setText("Your name is: " + fName + " " + lName);    }  } |

Находим TextView, затем получаем **Intent**и **извлекаем**из него String-объекты с именами *fname*и *lname*. Это те самые значения, которые мы помещали в коде MainActivity.java. Формируем строку вывода в TextView с использованием полученных данных.

Не забудьте прописать ViewActivity в манифесте. На этот раз никаких Intent Filter не нужно, т.к. мы точно знаем имя класса Activity и используем явный вызов.

**- вызываем Activity с возвратом результата**

Бывает необходимость **вызвать Activity**, **выполнить**на нем какое-либо **действие**и **вернуться** с **результатом**. Например – при создании SMS. Вы жмете кнопку «добавить адресата», система показывает экран со списком из адресной книги, вы выбираете нужного вам абонента и возвращаетесь в экран создания SMS. Т.е. вы **вызвали экран выбора**абонента, а он **вернул**вашему экрану **результат**.

Об этом можно почитать [здесь](http://developer.android.com/reference/android/app/Activity.html#StartingActivities) и [здесь](http://developer.android.com/guide/topics/fundamentals/activities.html#StartingAnActivityForResult).

Давайте посмотрим на практике. Создадим приложение с двумя экранами. С первого экрана будем вызывать второй экран, там вводить данные, нажимать кнопку и возвращаться на первый экран с введенными данными. Например, будем таким образом запрашивать имя.

# [**Урок 33. Хранение данных. Preferences.**](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/73-urok-33-hranenie-dannyh-preferences.html)

Хватит об Intent и Activity. Поговорим о хранении данных. В Android есть несколько способов хранения данных:

**Preferences** - в качестве аналогии можно привести виндовые INI-файлы

**SQLite** - база данных, таблицы

**обычные файлы** - внутренние и внешние (на SD карте)