[Task. Что это такое и как формируется](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/62-urok-25-task-chto-eto-takoe-i-kak-formiruetsja.html" \o "Урок 25. Task. Что это такое и как формируется)

В этом уроке:

- немного теории по Task  
- фиксируем Activity в состоянии Paused

В этом уроке узнаем, куда помещается Activity, пока его не видно. И откуда оно достается при нажатии кнопки назад. В [хелпе](http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html" \t "_blank) об этом написано достаточно понятно. Я сделаю краткий перевод основной части этого хелпа и использую их схемы.

Task

Мы уже знаем, что приложение может содержать несколько Activity. И что Activity умеет вызывать Activity из других приложений с помощью Intent и Intent Filter. Если вы хотите отправить письмо из вашего приложения, вы вызываете Activity почтовой программы и передаете ей данные. Письмо уходит и вы возвращаетесь в ваше приложение. Создается ощущение, что все это происходило в рамках одного приложения. Такая «бесшовность» достигается за счет того, что оба Activity (ваше и почтовое) были в одном Task.

*Прежде, чем продолжу объяснять, хочу сразу привести аналогию, чтобы тему легче было понять. В скобках я буду давать понятия-аналоги из Android.*

*Механизм организации Activity* *в Android* *очень схож по реализации с навигацией в браузере. Вы находитесь в одной вкладке(Task) и открываете страницы (Activity) переходя по ссылкам (Intent). В любой момент можете вернуться на предыдущую страницу, нажав кнопку Назад. Но кнопка Вперед отсутствует, т.к. страница, на которой была нажата кнопка Назад, стирается из памяти. И надо снова нажимать ссылку, если хотим попасть на нее. Если вам надо открыть что-то новое, вы создаете новую вкладку и теперь уже в ней открываете страницы, переходите по ссылкам, возвращаетесь назад. В итоге у вас есть несколько вкладок. Большинство из них на заднем фоне, а одна (активная, с которой сейчас работаете) – на переднем.*

*В итоге список аналогий браузера и Android таков:*

*Браузер – Android  
Вкладка с историей посещений – Task  
Страница – Activity  
Ссылка – Intent*

*Теперь вам будет более понятен текст про Task.*

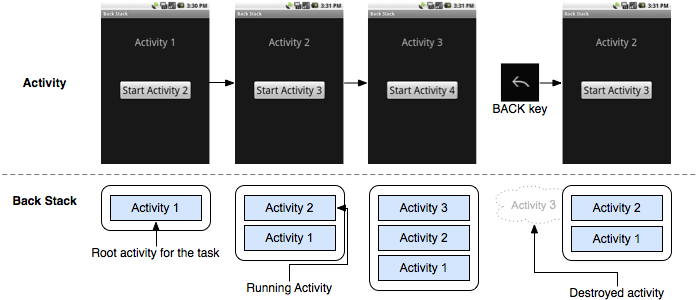
Task – группа из нескольких Activity, с помощью которых пользователь выполняет определенную операцию. Обычно стартовая позиция для создания Task – это экран Домой (Home).

Находясь в Home вы вызываете какое-либо приложение из списка приложений или через ярлык. Создается Task. И Activity приложения (которое отмечено как MAIN в манифест-файле) помещается в этот Task как корневое. Task выходит на передний фон. Если же при вызове приложения, система обнаружила, что в фоне уже существует Task, соответствующий этому приложению, то она выведет его на передний план и создавать ничего не будет.

Когда Activity\_A вызывает Activity\_B, то Activity\_B помещается на верх (в топ) Task и получает фокус. Activity\_A остается в Task, но находится в состоянии Stopped (его не видно и оно не в фокусе). Далее, если пользователь жмет Back находясь в Activity\_B, то Activity\_B удаляется из Task и уничтожается. А Activity\_A оказывается теперь на верху Task и получает фокус.

В каком порядке открывались (добавлялись в Task) Activity, в таком порядке они и содержатся в Task. Они никак специально не сортируются и не упорядочиваются внутри. Набор Activity в Task еще называют back stack. Я буду называть его просто - стэк.

Схема (с офиц.сайта) демонстрирует пример:



 В верхней части то, что видит пользователь. В нижней – содержимое Task. Видно, как при вызове новых Activity они добавляются в верх стэка. А если нажата кнопка Назад, то верхнее Activity из стэка удаляется и отображается предыдущее Activity.

Допустим у нас есть Task с несколькими Activity. Он на переднем фоне, мы с ним работаем сейчас.

- если мы нажмем кнопку Home, то ничего не будет удалено, все Activity сохранятся в этом Task-е, а сам Task просто уйдет на задний фон и его всегда можно будет вызвать оттуда, снова вызвав приложение, Activity которого является корневым для Task-а. Либо можно удерживать кнопку Home и мы увидим как раз список Task-ов, которые расположены на заднем фоне.

-  если же в активном Task-е несколько раз нажимать кнопку Назад, то в итоге в стэке не останется Activity, пустой Task будет удален и пользователь увидит экран Home.

Там еще как всегда куча нюансов и сложностей, но мы пока остановимся на этом и в дебри не полезем. Этих знаний вполне хватит, чтобы ответить на вопросы предыдущего урока: *почему на шаге 2 MainActivity исчезло с экрана, но осталось висеть в памяти и не было уничтожено? Ведь на шаге 3 было уничтожено ActivityTwo после того, как оно пропало с экрана. А на шаге 4 было в итоге уничтожено и MainActivity. Почему шаг 2 стал исключением?*

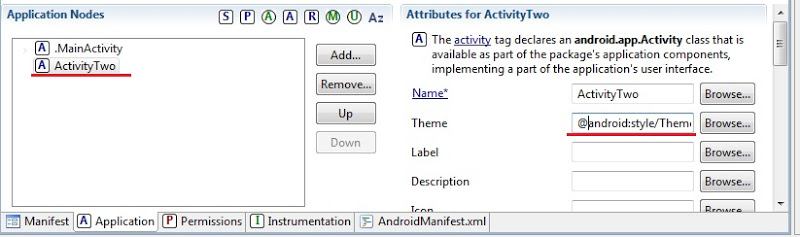
Теперь вы знаете, почему. Потому, что на шаге 2 MainActivity осталось в стэке, а ActivityTwo вставилось на верх стэка и получило фокус. Ну а на шаге 3 и 4 были удалены Activity из верха стэка, в Task не осталось Activity, и мы увидели экран Home.

Если бы мы на шаге 3 нажали не Back, а Home, то Task с обоими Activity ушел бы задний фон и ничего не было бы уничтожено.

Paused

Теперь давайте откроем проект с прошлого урока P0241\_TwoActivityState. Мы хотели поймать состояние Paused для Activity. Это состояние означает, что Activity не в фокусе, но оно видно, пусть и частично. Мы можем этого добиться, если присвоим диалоговый стиль для ActivityTwo. Оно отобразится как всплывающее окно и под ним будет частично видно MainActivity – оно и будет в статусе Paused. Давайте реализуем.

Для этого открываем AndroidManifest.xml, вкладка Application, находим там ActivityTwo и справа в поле Theme пишем такой текст: *@android:style/Theme.Dialog*



Все сохраняем и запускаем приложение.

Появилось MainActivity

Логи:

*MainActivity: onCreate()*  
*MainActivity: onStart()*  
*MainActivity: onResume()*

Все верно.

Вызываем ActivityTwo.

Логи:

*MainActivity: onPause()*  
*ActivityTwo: onCreate()*  
*ActivityTwo: onStart()*  
*ActivityTwo: onResume()*

Видим, что не был вызван метод onStop для MainActivity, а значит приложение не было переведено в состояние Stopped и находится в режиме Paused.

Нажмем Back.

Логи:

*ActivityTwo: onPause()*  
*MainActivity: onResume()*  
*ActivityTwo: onStop()*  
*ActivityTwo: onDestroy()*

MainActivity восстановилось одним лишь вызовом onResume, а onStart не понадобился, т.к. оно было в состоянии Paused, а не Stopped.

Мы четко увидели разницу между этим примером и им же на прошлом уроке. И MainActivity у нас был в состоянии Paused.

Далее можно нажать Back, а можно Home - вы уже знаете, что произойдет в обоих случаях. По логам можно убедиться в этом.

Чтобы вернуть ActivityTwo нормальный режим отображения, зайдите снова в манифест и удалите строку из поля Theme.

Кстати, у вас уже вполне достаточно знаний, чтобы создать приложение с кучей Activity, прописать вызовы и поиграться, посмотреть логи. Тем самым закрепите темы LifeCycle и Task.

[Урок 26. Intent Filter - практика](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/64-urok-26-intent-filter-praktika.html)

20 октября 2011

В этом уроке:

- вызываем Activity, используя неявный вызов и Intent Filter

Последние уроки получились перегруженными теорией. Эту теорию необходимо понять, чтобы не было проблем с практикой. Эти темы являются основой – Task, Lifecycle, Intent. Если что-либо осталось непонятно, то вы всегда можете снова открыть и перечитать материал. А далее мы будем реализовывать примеры, которые будут эту теорию подтверждать и все станет понятнее.

На прошлых уроках мы научились **вызывать Activity**с помощью **Intent**и явного указания **класса**. Также мы знаем, что есть и другой способ вызова Activity – **неявный**. Он основан на том, что Activity вызывается не по имени, а по функционалу. Т.е. мы хотим выполнить определенные действия, создаем и настраиваем соответствующий **Intent**и отправляем его **искать**те Activity, которые могли бы справиться с нашей задачей.

Давайте посмотрим, как это реализуется на практике. Мы создадим приложение, которое будет отображать нам текущее время или дату. Сделаем мы это с помощью трех Activity:  
- первое будет содержать две кнопки: Show time и Show date  
- второе будет отображать время  
- третье будет отображать дату

Нажатие на кнопку Show time будет вызывать второе Activity, а нажатие на кнопку Show date – третье Activity. Но **реализуем**мы это не через прямое указание классов Activity в Intent, а через**Intent Filter**.

Создадим проект:

**Project name**: P0261\_IntentFilter  
**Build Target**: Android 2.3.3  
**Application name**: IntentFilter  
**Package name**: ru.startandroid.develop.p0261intentfilter  
**Create Activity**: MainActivity

Открываем**main.xml** и рисуем две кнопки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="horizontal">      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/btnTime"          android:text="Show time">      </Button>      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/btnDate"          android:text="Show date">      </Button>  </LinearLayout> |

Пишем реализацию **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 | package ru.startandroid.develop.p0261intentfilter;    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;    public class MainActivity extends Activity implements OnClickListener {        /\*\* Called when the activity is first created. \*/      @Override      public void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.main);            Button btnTime = (Button) findViewById(R.id.btnTime);          Button btnDate = (Button) findViewById(R.id.btnDate);            btnTime.setOnClickListener(this);          btnDate.setOnClickListener(this);      }      @Override    public void onClick(View v) {      Intent intent;        switch(v.getId()) {      case R.id.btnTime:        intent = new Intent("ru.startandroid.intent.action.showtime");        startActivity(intent);        break;      case R.id.btnDate:        intent = new Intent("ru.startandroid.intent.action.showdate");        startActivity(intent);        break;      }    }  } |

В коде мы определили кнопки и присвоили им **Activity**как **обработчик**нажатий. В методе **onCilck**мы определяем какая кнопка была нажата и **создаем Intent**.

Для создания Intent используем конструктор: [Intent (String action)](http://developer.android.com/reference/android/content/Intent.html" \l "Intent(java.lang.String)" \t "_blank). Т.е. мы при создании заполняем атрибут объекта Intent, который называется **action**. Это обычная строковая константа. Action обычно указывает **действие**, которое мы хотим произвести. Например, есть следующие системные action-константы: [ACTION\_VIEW](http://developer.android.com/reference/android/content/Intent.html#ACTION_VIEW) - просмотр, [ACTION\_EDIT](http://developer.android.com/reference/android/content/Intent.html#ACTION_EDIT) – редактирование, [ACTION\_PICK](http://developer.android.com/reference/android/content/Intent.html#ACTION_PICK) – выбор из списка, [ACTION\_DIAL](http://developer.android.com/reference/android/content/Intent.html#ACTION_DIAL) – сделать звонок.

Если действие производится с чем-либо, то в пару к **action**идет еще один Intent-атрибут – **data**. В нем мы можем указать какой-либо объект: пользователь в адресной книге, координаты на карте, номер телефона и т.п. Т.е. **action**указывает **что делать**, а **data**– **с чем делать**.

Про **data**мы еще поговорим на следующих уроках, а пока будем использовать только **action**. Выше я уже перечислил некоторые системные action-константы, но мы можем использовать и свой action.

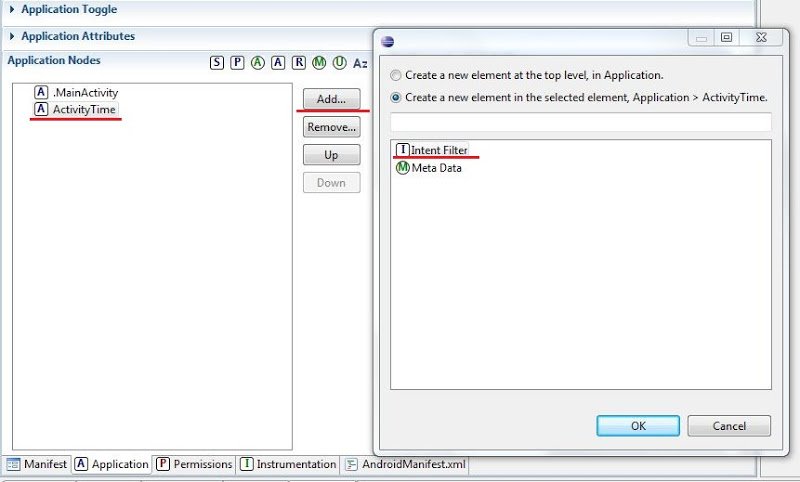
Как вы видите из кода, я придумал и использую такие action:  
*ru.startandroid.intent.action.showtime*  
*ru.startandroid.intent.action.showdate*

Первый будет означать, что я хочу вызвать Activity, которое мне покажет текущее время. Второй – Activity с датой.

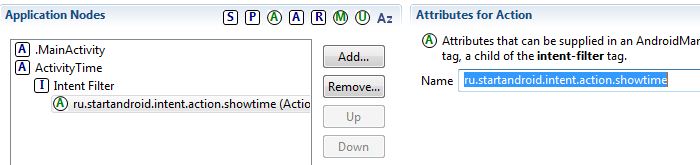
Здесь надо четко понимать следующее: **action**– это просто **текст**. И я мог с таким же успехом придумать action *abcdefg123456*. Но текст *showtime* – отражает то, что я хочу сделать, он нагляднее и понятнее. А префикс *ru.startandroid.intent.action* я использую, чтобы не было коллизий. В системе может быть приложение, которое уже использует action *showtime* - я не хочу с ним пересекаться. Поэтому мой action – это *ru.startandroid.intent.action.showtime*.

Итак, мы создали **Intent**с **action**и запустили его в систему искать Activity. Чтобы Activity подошла, надо чтобы ее **Intent Filter**содержал атрибут **action**с тем же **значением**, что и **action**в **Intent**. Значит нам осталось создать две **Activity**, настроить их**Intent Filter** и реализовать отображение времени и даты.

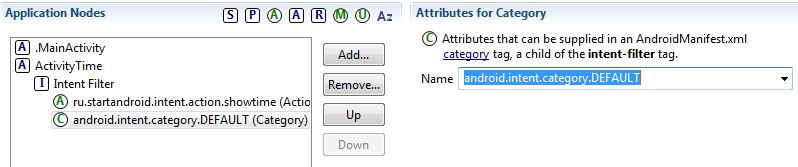
Activity создается как обычно - создаем класс **ActivityTime**с суперклассом **android.app.Activity** и прописываем его в манифесте как Activity. После того, как прописали в манифесте, надо будет там же создать Intent Filter. Для этого выделяем **ActivityTime**, жмем **Add**, выбираем**Intent Filter** и жмем **ОК**.



Далее в Intent Filter аналогично создаем **Action**и в поле **Name**прописываем *ru.startandroid.intent.action.showtime*



Также в **Intent Filter** надо создать **Category**и в поле **name**выбрать из списка *android.intent.category.DEFAULT*. Пока не будем разбираться детально зачем она нужна. Но без этого вызов **startActivity(Intent)** не найдет Activity.



Создадим layout для нового Activity, назовем его **time.xml**:

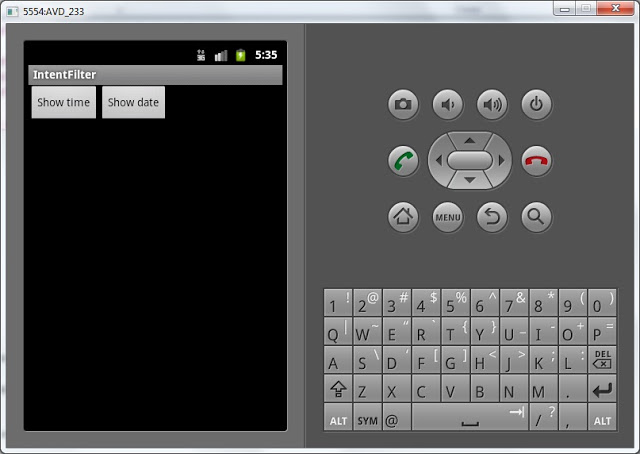
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="vertical">      <TextView          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/tvTime"          android:text="TextView"          android:layout\_gravity="center\_horizontal"          android:layout\_marginTop="20dp"          android:textSize="30sp">      </TextView>  </LinearLayout> |

Пишем код в **ActivityTime.java**:

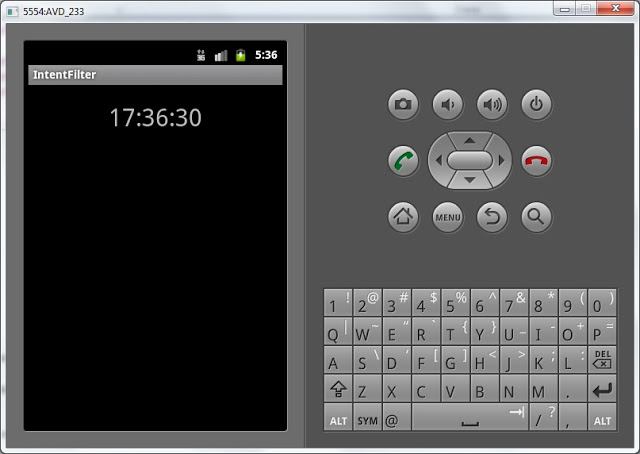
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package ru.startandroid.develop.p0261intentfilter;    import java.sql.Date;  import java.text.SimpleDateFormat;    import android.app.Activity;  import android.os.Bundle;  import android.widget.TextView;    public class ActivityTime extends Activity {      @Override    protected void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.time);        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");      String time = sdf.format(new Date(System.currentTimeMillis()));        TextView tvTime = (TextView) findViewById(R.id.tvTime);      tvTime.setText(time);    }  } |

Тут все просто - вычисляем текущее время и показываем его в TextView.

Все сохраним и запустим приложение.



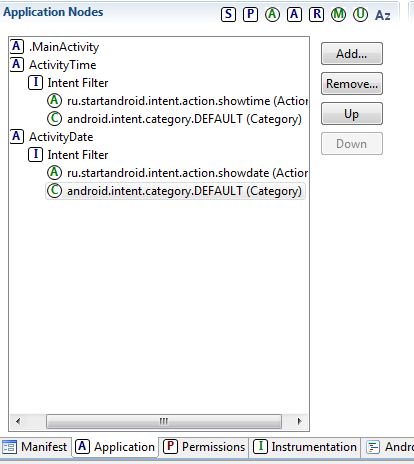
Жмем кнопку Show time:



отобразилось время. Т.е. **Intent**с **action**= *ru.startandroid.intent.action.showtime* нашел и отобразил **Activity**, у которого **action**также равен *ru.startandroid.intent.action.showtime* в **Intent Filter**.

Вернемся назад (кнопка **Back**) и нажмем теперь кнопку **Show date**. Приложение выдаст ошибку, т.к. оно не смогло найти Activity, которое соответствовало бы Intent с **action**= *ru.startandroid.intent.action.showdate* (мы создали только для showtime).

Давайте создадим такое Activity, назовем его **ActivityDate**. Действия все те же самые, что и при создании ActivityTime:  
- создание класса  
- создание Activity в манифесте и создание для него Intent Filter (c action = *ru.startandroid.intent.action.showdate* и category = *android.intent.category.DEFAULT*)



Layout-файл назовем **date.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:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/tvDate"          android:text="TextView"          android:layout\_gravity="center\_horizontal"          android:layout\_marginTop="20dp"          android:textSize="30sp">      </TextView>  </LinearLayout> |

Код **ActivityDate.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package ru.startandroid.develop.p0261intentfilter;    import java.sql.Date;  import java.text.SimpleDateFormat;    import android.app.Activity;  import android.os.Bundle;  import android.widget.TextView;    public class ActivityDate extends Activity {      @Override    protected void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.date);        SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");      String date = sdf.format(new Date(System.currentTimeMillis()));        TextView tvDate = (TextView) findViewById(R.id.tvDate);      tvDate.setText(date);    }  } |

Все сохраняем, запускаем приложение, жмем **Show date** и видим дату. Это значит, что **Intent**с **action** = *ru.startandroid.intent.action.showdate* нашел и отобразил **ActivityDate**подходящее ему по **Intent Filter**.

Чтобы закрепить тему, проделаем еще кое-что. Если помните, в [уроке №22](http://startandroid.ru/uroki/vse-uroki-spiskom/59-urok-22-intent-intent-filter-context-teorija.html), где я рассказывал про Intent Filter, я говорил, что **один Intent** может найти **несколько**подходящих **Activity**. В этом случае пользователю предоставляется **выбор**, какое Activity использовать. Давайте сами спровоцируем такой случай. Мы сделаем еще одно Activity, которое будет реагировать на Intent с action = *ru.startandroid.intent.action.showdate*. И будет отображать текущую дату аналогично ActivityDate. Но формат отображения даты будет немного другой.

Давайте создадим такое Activity, назовем его **ActivityDateEx**. Действия все те же самые, что и при создании ActivityDate:  
- создание класса  
- создание Activity в манифесте и создание для него Intent Filter (с action = *ru.startandroid.intent.action.showdate* и category = *android.intent.category.DEFAULT*)

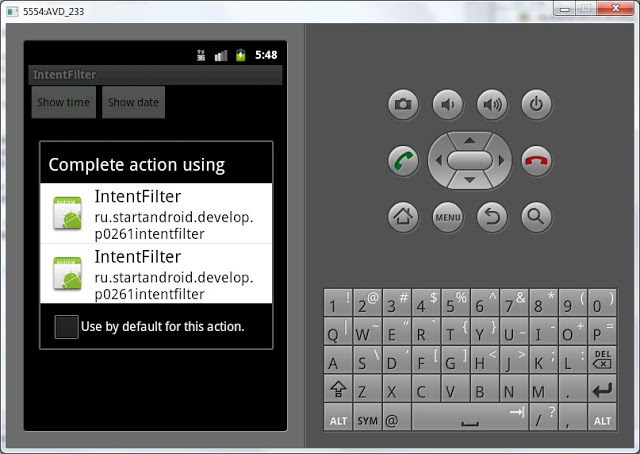
Новый layout-файл создавать не будем, используем уже существующий **date.xml**. В принципе, все три Activity у нас могли использовать один layout, т.к. они совершенно одинаковы – один TextView.

Код **ActivityDateEx.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package ru.startandroid.develop.p0261intentfilter;    import java.sql.Date;  import java.text.SimpleDateFormat;    import android.app.Activity;  import android.os.Bundle;  import android.widget.TextView;    public class ActivityDateEx extends Activity {      @Override    protected void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.date);        SimpleDateFormat sdf = new SimpleDateFormat("EEE, MMM d, yyyy");      String date = sdf.format(new Date(System.currentTimeMillis()));        TextView tvDate = (TextView) findViewById(R.id.tvDate);      tvDate.setText(date);    }  } |

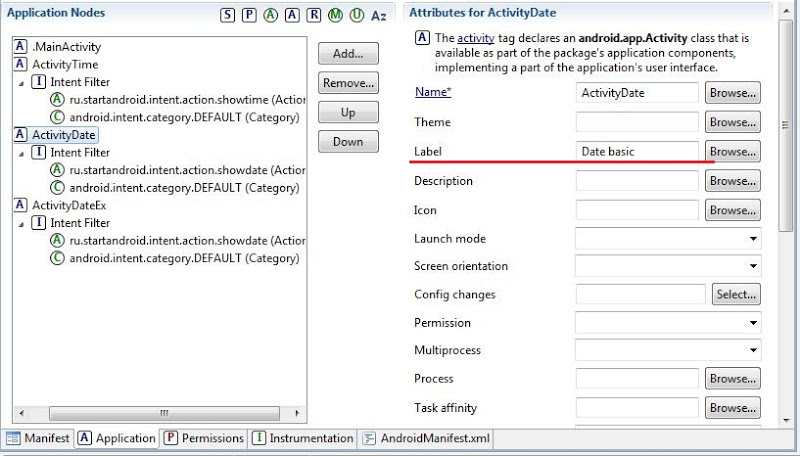
Как видим, отличие от ActivityDate только в формате даты.

Сохраним все и запустим. Жмем Show date и видим такой выбор:



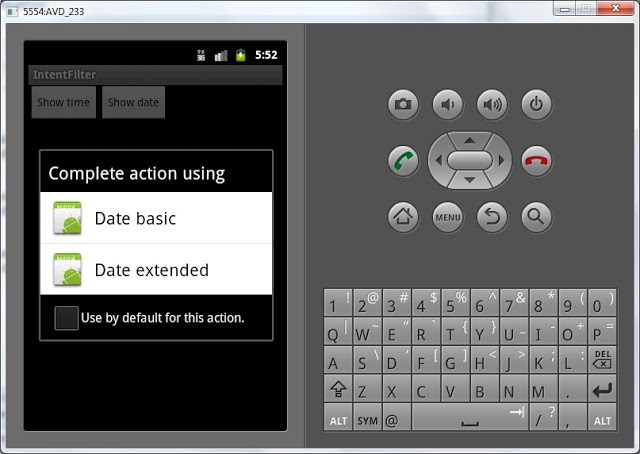
**Intent**нашел **два Activity**, но показал для каждого из них название родительского приложения и package. В нашем случае – оба Activity из нашего приложения, поэтому текст одинаков и не разберешь, какое из них какое. Давайте пофиксим это, прописав нормальные имена.

Нажмите Back, чтобы закрыть диалог выбора. Идем в **манифест**и для **Activity**пропишем **label**:



*Date basic* для **ActivityDate**  
*Date extended* для **ActivityDateEx**

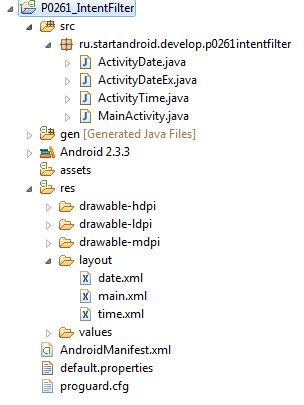
Сохраняем и запускаем. Жмем Show date и видим такой выбор:



Так значительно лучше. Жмем Date extended и видим дату в расширенном формате на ActivityDateEx.

Итак, мы создавали и посылали **Intent** с **action**. Этот **Intent**находил **Activity**с подходящим **Intent Filter**и отображал его. Если находил **несколько**– давал **выбор**. Примеры отлично показывают механизм.

Если запутались, чего и где создавать, привожу скрин проекта и содержимое манифеста.



Содержимое манифеста (вкладка AndroidManifest.xml):

|  |  |
| --- | --- |
| 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 | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="ru.startandroid.develop.p0261intentfilter" android:versionCode="1" android:versionName="1.0">      <uses-sdk android:minSdkVersion="10"></uses-sdk>      <application android:icon="@drawable/icon" android:label="@string/app\_name">          <activity android:name=".MainActivity" android:label="@string/app\_name">              <intent-filter>                  <action android:name="android.intent.action.MAIN"></action>                  <category android:name="android.intent.category.LAUNCHER"></category>              </intent-filter>          </activity>          <activity android:name="ActivityTime">              <intent-filter>                  <action android:name="ru.startandroid.intent.action.showtime"></action>                  <category android:name="android.intent.category.DEFAULT"></category>              </intent-filter>          </activity>          <activity android:name="ActivityDate" android:label="Date basic">              <intent-filter>                  <action android:name="ru.startandroid.intent.action.showdate"></action>                  <category android:name="android.intent.category.DEFAULT"></category>              </intent-filter>          </activity>          <activity android:name="ActivityDateEx" android:label="Date extended">              <intent-filter>                  <action android:name="ru.startandroid.intent.action.showdate"></action>                  <category android:name="android.intent.category.DEFAULT"></category>              </intent-filter>          </activity>      </application>  </manifest> |

[Читаем action из Intent](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/65-urok-27-chitaem-action-iz-intent.html)

24 октября 2011

В этом уроке:

- читаем action из Intent

На прошлом уроке мы сделали наглядный пример, показывающий, как связаны между собой **action**, **Intent**и **Intent** **Filter**. На этом уроке продолжим тему. Есть две новости: хорошая и хорошая )

Первая новость. **Intent Filter** может содержать в себе несколько **action**. Тем самым Activity дает понять, что она способна на **несколько действий**. Например, не только просмотр картинки, но и редактирование. Получается, что Activity может подойти разным Intent с разными action.

Вторая новость. **Activity**, которое было вызвано с помощью Intent, имеет **доступ**к этому **Intent**и может **прочесть**его **атрибуты**. Т.е. может узнать какой action использовался.

Мы сделаем следующее: создадим **Activity**и настроим**Intent Filter**на **action**= *ru.startandroid.intent.action.showtime* и **action**= *ru.startandroid.intent.action.showdate*. Тем самым мы обозначаем, что это Activity способно и время показать, и дату. Далее мы будем создавать Intent либо с action = *ru.startandroid.intent.action.showtime*, либо с *ru.startandroid.intent.action.showdate*. Они **оба**будут вызывать **одно Activity**. А чтобы **Activity**знало показывать ему дату или время, мы будем **читать** **action**из **Intent** и по нему определять.

В общем, сейчас начнем делать и все станет понятно )

Создадим проект:

**Project name**: P0271\_GetIntentAction  
**Build Target**: Android 2.3.3  
**Application name**: GetIntentAction  
**Package name**: ru.startandroid.develop.p0271getintentaction  
**Create Activity**: MainActivity

Открываем **main.xml** и рисуем две кнопки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="horizontal">      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/btnTime"          android:text="Show time">      </Button>      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/btnDate"          android:text="Show date">      </Button>  </LinearLayout> |

Экран точно такой же как и в прошлом уроке.

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 | package ru.startandroid.develop.p0271getintentaction;  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;    public class MainActivity extends Activity implements OnClickListener {        /\*\* Called when the activity is first created. \*/      @Override      public void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.main);            Button btnTime = (Button) findViewById(R.id.btnTime);          Button btnDate = (Button) findViewById(R.id.btnDate);            btnTime.setOnClickListener(this);          btnDate.setOnClickListener(this);      }      @Override    public void onClick(View v) {      Intent intent;        switch(v.getId()) {      case R.id.btnTime:        intent = new Intent("ru.startandroid.intent.action.showtime");        startActivity(intent);        break;      case R.id.btnDate:        intent = new Intent("ru.startandroid.intent.action.showdate");        startActivity(intent);        break;      }    }  } |

Код тоже полностью из прошлого урока. Определяем кнопки, присваиваем **обработчик**– **Activity**, и вызываем **Intent**по нажатиям. Теперь мы сделаем Activity, которая будет ловить оба этих Intent.

Для начала создадим layout-файл **info.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:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:id="@+id/tvInfo"          android:text="TextView"          android:layout\_gravity="center\_horizontal"          android:layout\_marginTop="20dp"          android:textSize="30sp">      </TextView>  </LinearLayout> |

На экране у нас один TextView.

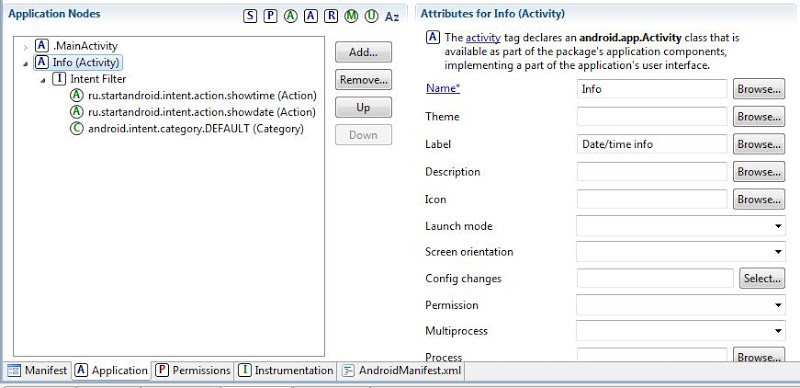
Создаем **Activity**, назовем его просто **Info**.

Код **Info.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  42  43 | package ru.startandroid.develop.p0271getintentaction;    import java.sql.Date;  import java.text.SimpleDateFormat;    import android.app.Activity;  import android.content.Intent;  import android.os.Bundle;  import android.widget.TextView;    public class Info extends Activity {      @Override    protected void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.info);        // получаем Intent, который вызывал это Activity      Intent intent = getIntent();      // читаем из него action      String action = intent.getAction();        String format = "", textInfo = "";        // в зависимости от action заполняем переменные      if (action.equals("ru.startandroid.intent.action.showtime")) {        format = "HH:mm:ss";        textInfo = "Time: ";      }      else if (action.equals("ru.startandroid.intent.action.showdate")) {        format = "dd.MM.yyyy";        textInfo = "Date: ";      }        // в зависимости от содержимого переменной format      // получаем дату или время в переменную datetime      SimpleDateFormat sdf = new SimpleDateFormat(format);      String datetime = sdf.format(new Date(System.currentTimeMillis()));        TextView tvDate = (TextView) findViewById(R.id.tvInfo);      tvDate.setText(textInfo + datetime);    }  } |

Мы получаем **Intent**с помощью метода [getIntent()](http://developer.android.com/reference/android/app/Activity.html" \l "getIntent()" \t "_blank), [читаем](http://developer.android.com/reference/android/content/Intent.html#getAction()) из него **action**и в зависимости от значения формируем и выводим на экран текст.

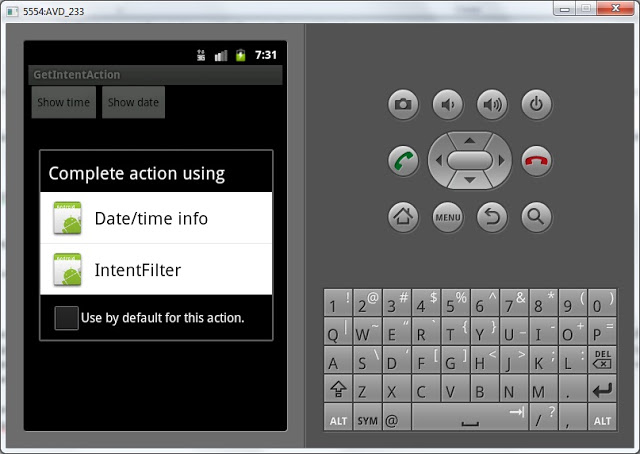
Не забываем прописать новое Activity в **манифесте**и создать ему **Intent Filter**с двумя **action**и **category**. И **label**укажите Date/time info.



**Intent Filter**для Info содержит **два** **action**. А значит если придет **Intent**c **любым**из них – то **Activity**будет **вызвана**.

Все сохраним и запустим.

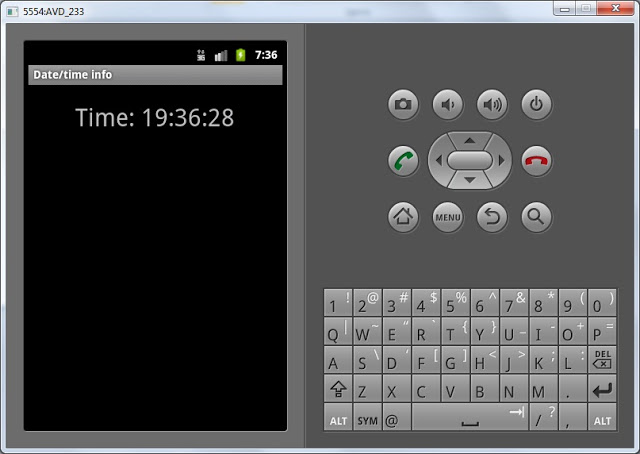
Жмем кнопку Show time. Скорее всего вы увидите следующее:



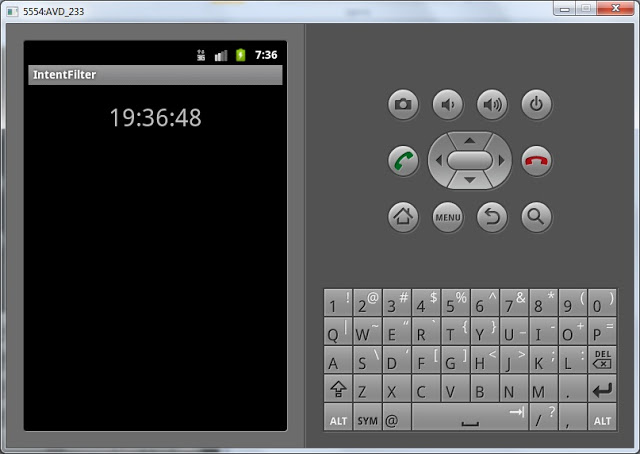
Система предлагает нам выбор. Т.е. **Intent**с action = *ru.startandroid.intent.action.showtime* нашел **два** подходящих **Activity**. То, которое *Date/time info* мы создали только что, тут все понятно. Вопрос – откуда второе с названием *IntentFilter*. Ответ – это ActivityTime из прошлого урока. Называется оно *IntentFilter*потому, что на прошлом уроке мы не прописали в манифесте  **label**для этого Activity и система по умолчанию отображает название приложения.

Если же система не отобразила диалог выбора, значит вы, либо не создавали приложение с прошлого урока, либо пересоздали AVD, либо где-то очепятка в коде.

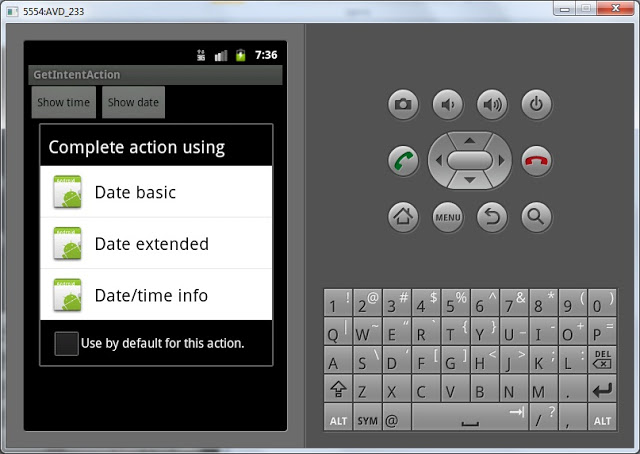
Выбираем *Date/time info* и видим то, что только что кодили. **Activity**определило, что **Intent**был с **action**= *ru.startandroid.intent.action.showtime* и показало время с текстом *Time:*



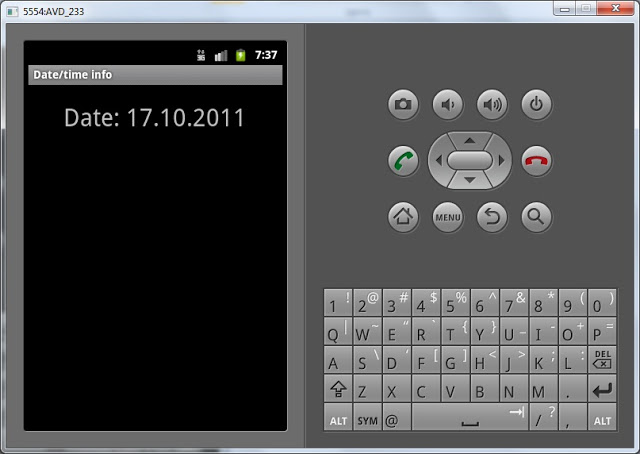
Если же выбрать *IntentFilter*увидим Activity с прошлого урока:



Теперь попробуем нажать кнопку Show date: видим такой выбор:

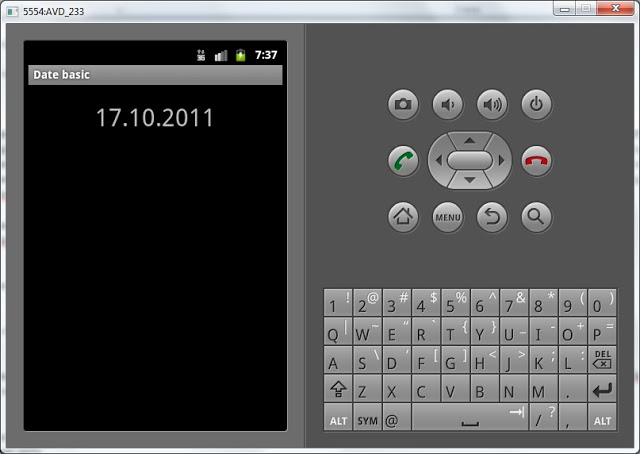


Снова видим наше *Date/time info* и **два Activity**с **прошлого**урока. Они все содержат **action**= *ru.startandroid.intent.action.showdate* в **Intent Filter** и нам надо выбирать. Выберем *Date/time info* и видим дату с текстом *Date:*

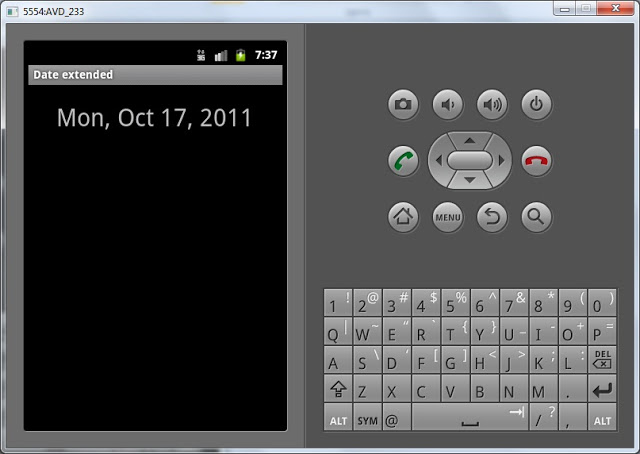


Если же выбирать *Date basic* или *Date extended* увидим то, что делали на прошлом уроке :

обычную дату



и расширенную



Мы увидели, что **одно Activity**может быть вызвано с помощью **Intent**с **разными action**. **Activity**может прочесть **action**и выполнить необходимые действия.

Также мы убедились, что **Intent**ищет **Activity**по **всем приложениям** в системе. В этот раз он нашел **Activity**из приложения, которое мы делали на прошлом уроке.

[Урок 28. Extras - передаем данные с помощью Intent](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/67-urok-28-extras-peredaem-dannye-s-pomoschju-intent.html)

27 октября 2011

В этом уроке:

- передаем данные с помощью Intent

На прошлых уроках мы узнали, что такое **Intent**и как им пользоваться. Из одного Activity мы просто вызывали другое, передавая **action**. Теперь научимся передавать **данные**. Сделаем простейшее приложение. На первом экране мы будем вводить наше имя и фамилию, а второй экран будет эти данные отображать. Передавать **данные**будем **внутри Intent**.

**Создадим** **проект:**

**Project name**: P0281\_IntentExtras  
**Build Target**: Android 2.3.3  
**Application name**: IntentExtras  
**Package name**: ru.startandroid.develop.p0281intentextras  
**Create Activity**: MainActivity

 Открываем**main.xml** и рисуем экран с полями и кнопкой отправки:

|  |  |
| --- | --- |
| 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  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical">      <TextView          android:layout\_width="fill\_parent"          android:layout\_height="wrap\_content"          android:layout\_marginTop="10dp"          android:gravity="center\_horizontal"          android:text="Input your Name">      </TextView>      <TableLayout          android:id="@+id/tableLayout1"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_margin="10dp"          android:stretchColumns="1">          <TableRow              android:id="@+id/tableRow1"              android:layout\_width="match\_parent"              android:layout\_height="wrap\_content">              <TextView                  android:id="@+id/textView1"                  android:layout\_width="match\_parent"                  android:layout\_height="wrap\_content"                  android:text="First Name">              </TextView>              <EditText                  android:id="@+id/etFName"                  android:layout\_width="match\_parent"                  android:layout\_height="wrap\_content"                  android:layout\_marginLeft="5dp">                  <requestFocus>                  </requestFocus>              </EditText>          </TableRow>          <TableRow              android:id="@+id/tableRow2"              android:layout\_width="match\_parent"              android:layout\_height="wrap\_content">              <TextView                  android:id="@+id/textView2"                  android:layout\_width="match\_parent"                  android:layout\_height="wrap\_content"                  android:text="Last Name">              </TextView>              <EditText                  android:id="@+id/etLName"                  android:layout\_width="match\_parent"                  android:layout\_height="wrap\_content"                  android:layout\_marginLeft="5dp">              </EditText>          </TableRow>      </TableLayout>      <Button          android:id="@+id/btnSubmit"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_gravity="center\_horizontal"          android:text="Submit">      </Button>  </LinearLayout> |

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" \l "putExtra(java.lang.String,%20java.lang.CharSequence)" \t "_blank). 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:

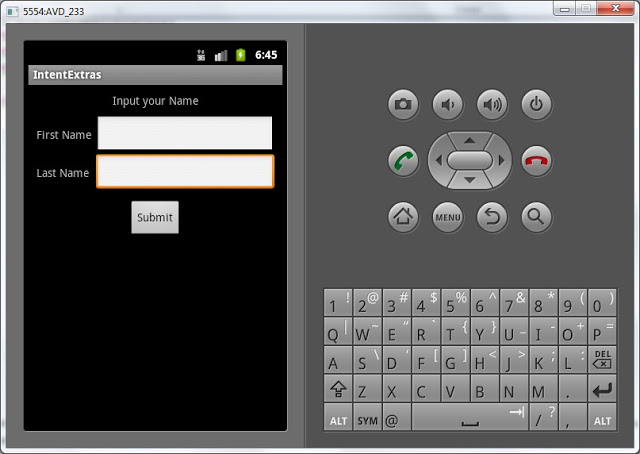
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package ru.startandroid.develop.p0281intentextras;    import android.app.Activity;  import android.content.Intent;  import android.os.Bundle;  import android.widget.TextView;    public 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.

Все сохраним и запустим.

Видим такой экран:



Заполняете поля как пожелаете. Я напишу *John*в поле **First Name** и *Smith* в поле**Last Name**.

Жмем Submit:



ViewActivity отобразилось, считало данные из Intent и вывело их на экран.

Поместить в Intent можно данные не только типа String. В [списке методов](http://developer.android.com/reference/android/content/Intent.html#pubmethods) Intent можно посмотреть все многообразие типов, которые умеет принимать на вход метод putExtra.