# Переключение между экранами приложения

## 

## Простое переключение на другой экран

Приложение не всегда состоит из одного экрана. Например, мы создали очень полезную программу и пользователю хочется узнать, кто же её автор. Он нажимает на кнопку «О программе» и попадает на новый экран, где находится полезная информация о версии программы, авторе, адресе сайта, сколько у автора котов и т.д. Воспринимайте экран активности как веб-страницу с ссылкой на другую страницу. Если вы посмотрите на код в файле **MainActivity.java** из прошлых уроков, то увидите, что наш класс **MainActivity** тоже относится к **Activity** (или его наследникам) или, если говорить точнее, наследуется от него.

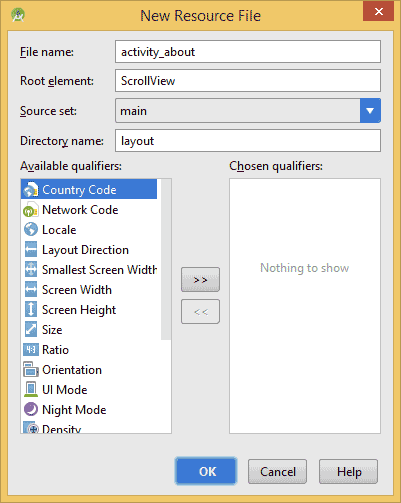
public class MainActivity extends AppCompatActivity

Как нетрудно догадаться, нам следует создать новый класс, который может быть похож на **MainActivity** и затем как-то переключиться на него при нажатии кнопки.

Для эксперимента мы возьмем программу из первого урока и будем использовать для опытов кнопку (или создайте новый проект с одной кнопкой на экране). Далее создадим новую форму для отображения полезной информации. Например, покажем пользователю, что делает кот, когда идёт налево и направо. Согласитесь, это очень важная информация, дающая ключ к разгадке Вселенной.

Создавать новую активность будем вручную, хотя в студии есть готовые шаблоны. Но там ничего сложного и для лучшего понимания полезно всё делать руками.

Создадим новый XML-файл разметки **activity\_about.xml** в папке **res/layout**. Щёлкните правой кнопкой мыши на папке **layout** и выберите из контекстного меню **New | Layout resource file**. Появится диалоговое окно. В первом поле вводим имя файла **activity\_about**. Во втором нужно ввести корневой элемент. По умолчанию там стоит **ConstraintLayout**. Стираем текст и вводим **ScrollView**. Ввода нескольких символов достаточно, чтобы студия подсказала готовые варианты, можно сразу нажать Enter, не дожидаясь полного ввода слова:



Получится соответствующая заготовка, в которую вставим элемент **TextView**.

<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView\_about\_content"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/about\_text"/>

</ScrollView>

Информация будет извлекаться из ресурсов, а именно из строкового ресурса **about\_text**. Сейчас он подсвечен красным цветом, сигнализируя об отсутствии информации. Можно было нажать **Alt+Enter** и ввести текст в диалоговом окне. Но для нашего примера этот способ не подойдёт, так как наш текст будет многострочным, с использованием управляющих символов. Поэтому поступим по-другому. Откроем файл **res/values/strings.xml** и вводим следующий текст вручную:

<string name="about\_text">

У лукоморья дуб зелёный;\n

Златая цепь на дубе том:\n

И днём и ночью <b>кот учёный</b>\n

Всё ходит по цепи кругом;\n

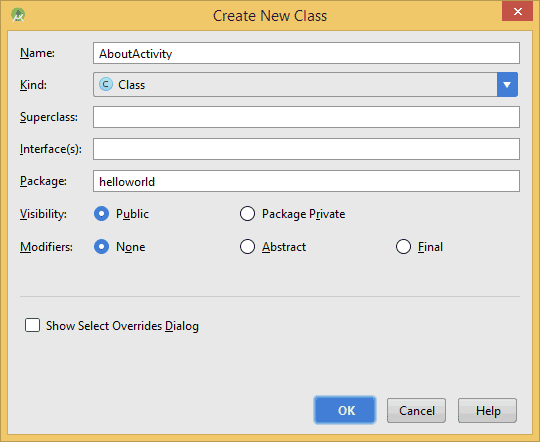
Идёт <b>направо</b> - песнь заводит,\n

<b>Налево</b> - сказку говорит.</string>

Мы использовали простейшие HTML-теги форматирования текста типа <b>, <i>, <u>. Для нашего примера достаточно выделить жирным слова, которые относятся к коту и направлению движения. Для перевода текста на новую строку используйте символы **\n**. Добавим ещё один строковый ресурс для заголовка нового экрана:

<string name="about\_title">О программе</string>

С разметкой разобрались. Далее необходимо создать класс для окна **AboutActivity.java**. Выбираем в меню **File | New | Java Class** и заполняем нужные поля. На первых порах достаточно указать только имя. Потом разберётесь с другими полями.



Получим заготовку.

Сейчас класс практически пустой. Добавим код вручную. Класс должен наследоваться от абстрактного класса **Activity** или его родственников типа **FragmentActivity**, **AppCompatActivity** и т.д. Дописываем **extends Activity**. У класса активности должен быть метод **onCreate()**. Ставим курсор мыши внутри класса и выбираем в меню **Code | Override Methods** (Ctrl+O). В диалоговом окне ищем нужный класс, можно набирать на клавиатуре первые символы для быстрого поиска. В созданном методе нужно вызвать метод **setContentView()**, который подгрузит на экран подготовленную разметку. У нас получится такой вариант.

package ru.alexanderklimov.helloworld;

import android.app.Activity;

import android.os.Bundle;

/\*\*

\* Created by Alexander Klimov on 01.12.2014.

\*/

public class AboutActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_about);

}

}

Теперь начинается самое главное. Наша задача - перейти на новый экран при щелчку кнопки на первом экране. Переходим обратно к классу **MainActivity**. Напишем обработчик щелчка кнопки:

public void onClick(View view) {

Intent intent = new Intent(MainActivity.this, AboutActivity.class);

startActivity(intent);

}

Здесь я использовал способ обработки нажатия кнопки, о котором рассказывалось в занятии [Щелчок кнопки/Счётчик ворон](http://developer.alexanderklimov.ru/android/android3.php).

Для запуска нового экрана необходимо создать экземпляр класса **Intent** и указать в первом параметре текущий класс, а во втором - класс для перехода, у нас это **AboutActivity**. После этого вызывается метод **startActivity()**, который и запускает новый экран.

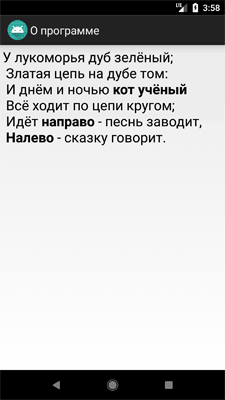
Если вы сейчас попытаетесь проверить работу приложения в эмуляторе, то получите сообщение об ошибке. Что мы сделали неправильно? Мы пропустили один важный шаг. Необходимо зарегистрировать новый **Activity** в манифесте **AndroidManifest.xml**. Найдите этот файл в своем проекте и дважды щёлкните на нём. Откроется окно редактирования файла. Добавьте новый тег **<activity>** после закрывающего тега **</activity>** для первой активности. Печатайте самостоятельно и активно используйте подсказки. Получится следующее:

<activity android:name=".AboutActivity"

android:label="@string/about\_title">

</activity>

Вот и пригодился строковый ресурс **about\_title**. Запускаем приложение, щёлкаем на кнопке и получаем окно **О программе**. Таким образом мы научились создавать новое окно и вызывать его по щелчку кнопки. А в нашем распоряжении появилась мегаудобная программа - теперь всегда под рукой будет подсказка, что делает кот, когда идёт налево.



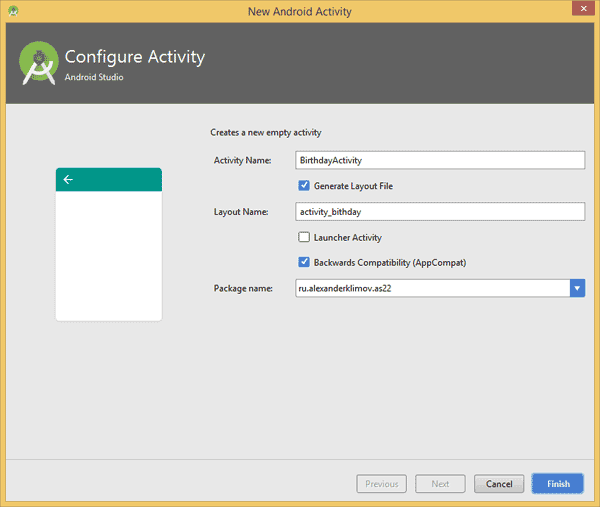
Ещё раз обращаю внимание, что второй создаваемый класс активности должен наследоваться от класса **Activity** или ему похожих (**ListActivity** и др.), иметь XML-файл разметки (если требуется) и быть прописан в манифесте.

После вызова метода **startActivity()** запустится новая активность (в данном случае **AboutActivity**), она станет видимой и переместится на вершину стека, содержащего работающие компоненты. При вызове метода **finish()** из новой активности (или при нажатии аппаратной клавиши возврата) она будет закрыта и удалена из стека. Разработчик также может перемещаться к предыдущей (или к любой другой) активности, используя всё тот же метод **startActivity()**.

## Создаём третий экран - способ для ленивых

Программисты, как и коты, существа ленивые. Постоянно помнить, что для активности нужно создать разметку и класс, который наследуется от **Activity**, а затем не забыть прописать класс в манифесте - да ну нафиг.

В этом случае выберите в меню **File | New | Activity | Basic Activity** (или другой шаблон). Дальше появится знакомое вам окно создания новой активности. Заполняем необходимые поля.



Нажимаем на кнопку **Finish** и активность будет готова. Чтобы убедиться в этом, откройте файл манифеста и проверьте наличие новой записи. Про файлы класса и разметки я уже не говорю, они сами появятся перед вами.

Самостоятельно добавьте новую кнопку на экране главной активности и напишите код для перехода на созданную активность.

На первых порах я бы посоветовал вам вручную создавать все необходимые компоненты для новой активности, чтобы понимать взаимосвязь между классом, разметкой и манифестом. А когда набьёте руку, то можете использовать мастер создания активности для ускорения работы.

## Передача данных между активностями

Мы использовали простейший пример для вызова другого экрана активности. Иногда требуется не только вызвать новый экран, но и передать в него данные. Например, имя пользователя. В этом случае нужно задействовать специальную область **extraData**, который имеется у класса **Intent**.

Область **extraData** - это список пар *ключ/значение*, который передаётся вместе с намерением. В качестве ключей используются строки, а для значений можно использовать любые примитивные типы данных, массивы примитивов, объекты класса **Bundle** и др.

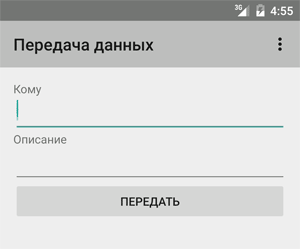
Для передачи данных в другую активность используется метод **putExtra()**:

intent.putExtra("Ключ", "Значение");

Принимающая активность должна вызвать какой-нибудь подходящий метод: **getIntExtra()**, **getStringExtra()** и т.д.:

int count = getIntent().getIntExtra("name", 0);

Переделаем предыдущий пример. У нас уже есть три активности. У первой активности разместим два текстовых поля и кнопку. Внешний вид может быть следующим:



У второй активности **SecondActivity** установим элемент **TextView**, в котором будем выводить текст, полученный от первой активности. Напишем следующий код для метода **onCreate()** у второй активности.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

String user = "ЖЫвотное";

String gift = "дырку от бублика";

TextView infoTextView = (TextView)findViewById(R.id.textViewInfo);

infoTextView.setText(user + " , вам передали " + gift);

}

Если сейчас запустить программу и просто вызвать второе окно, как это было описано в первой части статьи, то мы увидим надпись по умолчанию **ЖЫвотное, вам передали дырку от бублика**. Согласитесь, довольно обидно получать такие сообщения.

Исправляем ситуацию. Добавляем код у первой активности:

public void onClick(View view) {

EditText userEditText = (EditText) findViewById(R.id.editTextUser);

EditText giftEditText = (EditText) findViewById(R.id.editTextGift);

Intent intent = new Intent(MainActivity.this, SecondActivity.class);

// в ключ username пихаем текст из первого текстового поля

intent.putExtra("username", userEditText.getText().toString());

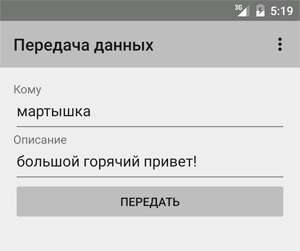
// в ключ gift пихаем текст из второго текстового поля

intent.putExtra("gift", giftEditText.getText().toString());

startActivity(intent);

}

Мы поместили в специальный контейнер объекта **Intent** два ключа со значениями, которые берутся из текстовых полей. Когда пользователь введёт данные в текстовые поля, они попадут в этот контейнер и будут переданы второй активности.



Вторая активность должна быть готова к тёплому приёму сообщений следующим образом (выделено жирным).

// Значения по умолчанию

String user = "ЖЫвотное";

String gift = "дырку от бублика";

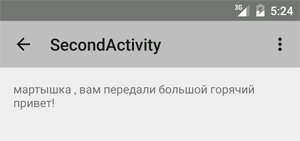
**user = getIntent().getExtras().getString("username");**

**gift = getIntent().getExtras().getString("gift");**

TextView infoTextView = (TextView)findViewById(R.id.textViewInfo);

infoTextView.setText(user + " , вам передали " + gift);

Теперь сообщение выглядит не столь обидным, а даже приятным для кое-кого. В сложных примерах желательно добавить проверку при обработке данных. Возможны ситуации, когда вы запустите вторую активность с пустыми данными типа **null**, что может привести к краху приложения.



В нашем случае мы знаем, что ждём строковое значение, поэтому код можно переписать так:

Intent intent = getIntent();

user = intent.getStringExtra("username");

Или так:

user = getIntent().getStringExtra("username");

У программы есть недостаток - не понятно, от кого мы получаем приветы. Любая хорошо воспитанная мартышка не возьмет подарок от анонимного источника. Поэтому в качестве домашнего задания добавьте ещё одно текстовое поле для ввода имени пользователя, который отправляет сообщение.



Google рекомендует для ключей использовать следующий формат: имя вашего пакета в качестве префикса, а затем сам ключ. В этом случае можно быть уверенным в уникальности ключа при взаимодействии с другими приложениями. Приблизительно так:

public final static String USER = "ru.alexanderklimov.myapp.USER";

## Кто подставил кота Ваську - получаем результат обратно

Не всегда бывает достаточно просто передать данные другой активности. Иногда требуется получить информацию обратно от другой активности при её закрытии. Если раньше мы использовали метод **startActivity(Intent intent)**, то существует родственный ему метод **startActivityForResult(Intent intent, int RequestCode)**. Разница между методами заключается в дополнительном параметре **RequestCode**. По сути это просто целое число, которое вы можете сами придумать. Оно нужно для того, чтобы различать от кого пришёл результат. Допустим у вас есть пять дополнительных экранов и вы присваиваете им значения от 1 до 5, и по этому коду вы сможете определить, чей результат вам нужно обрабатывать. Вы можете использовать значение -1, тогда это будет равносильно вызову метода **startActivity()**, т.е. никакого результата не получим.

Если вы используете метод **startActivityForResult()**, то вам необходимо переопределить в коде метод для приёма результата **onActivityResult()** и обработать полученный результат. Запутались? Давайте разберём пример.

Предположим, вы сыщик. Поступила информация, что в ресторане со стола влиятельного человека украли два кусочка колбасы и другие продукты. Подозрение пало на трёх подозреваемых - ворона, сраный пёсик и кот Васька.

Один из посетителей предоставил серию фотографий со своего понтового айфона:

Также имеются показания другого свидетеля: *А Васька слушает, да ест*.

Создаём новый проект **Sherlock** с двумя активностями. На первом экране будет кнопка для переключения на второй экран и текстовая метка, в которой будет отображено имя воришки.

<?xml version="1.0" encoding="UTF-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/LinearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="5dp" >

<ImageView

android:id="@+id/icon"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:adjustViewBounds="true"

android:background="#fa2255"

android:scaleType="centerCrop"

android:src="@drawable/sherlock" />

<TextView

android:id="@+id/textViewLabel"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/who"

android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView

android:id="@+id/textViewInfo"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge" />

<Button

android:id="@+id/buttonChoose"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Сделать выбор" />

</LinearLayout>

На втором экране будет группа переключателей:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/LinearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="10dip" >

<TextView

android:id="@+id/textView1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Выберите правильный ответ"

android:textAppearance="?android:attr/textAppearanceMedium" />

<RadioGroup

android:id="@+id/radioGroup1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" >

<RadioButton

android:id="@+id/radioCrow"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="false"

android:onClick="onRadioClick"

android:text="Ворона" />

<RadioButton

android:id="@+id/radioDog"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onRadioClick"

android:text="Сраный пёсик" />

<RadioButton

android:id="@+id/radioCat"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onRadioClick"

android:text="Кот Васька" />

</RadioGroup>

</LinearLayout>

Так как мы будем ожидать ответ из второго экрана, то нам необходимо задействовать метод **startActivityForResult()** на первом экране, в котором мы передадим переменную **CHOOSE\_THIEF** в качестве параметра **RequestCode**.

static final private int CHOOSE\_THIEF = 0;

public void onClick(View v) {

Intent questionIntent = new Intent(MainActivity.this,

ChooseActivity.class);

startActivityForResult(questionIntent, CHOOSE\_THIEF);

}

Посмотрите на код. При щелчке на кнопке мы собираемся работать со вторым экраном **ChooseActivity** и запускаем второй экран с ожиданием результата.

Переходим на второй экран и будем писать код для второй активности.

public final static String THIEF = "ru.alexanderklimov.sherlock.THIEF";

public void onRadioClick(View v) {

Intent answerIntent = new Intent();

switch (v.getId()) {

case R.id.radioDog:

answerIntent.putExtra(THIEF, "Сраный пёсик");

break;

case R.id.radioCrow:

answerIntent.putExtra(THIEF, "Ворона");

break;

case R.id.radioCat:

answerIntent.putExtra(THIEF, "Лошадь Пржевальского");

break;

default:

break;

}

setResult(RESULT\_OK, answerIntent);

finish();

}

Здесь всё просто, когда сыщик выбирает имя преступника, то через метод **putExtra()** мы передаём имя ключа и его значение.

Для удобства, после выбора мы сразу закрываем второе окно и перед закрытием передаём значение **RESULT\_OK**, чтобы было понятно, что выбор сделан. Если пользователь закроет экран через кнопку Back, то будет передано значение **RESULT\_CANCELED**.

Метод **setResult()** принимает два параметра: результирующий код и сам результат, представленный в виде намерения. Результирующий код говорит о том, с каким результатом завершилась работа активности, как правило, это либо **Activity.RESULT\_OK**, либо **Activity.RESULT\_CANCELED**. В некоторых случаях нужно использовать собственный код возврата для обработки специфических для вашего приложения вариантов. Метод **setResult()** поддерживает любое целочисленное значение.

Если вы будете передавать данные явно через кнопку, то неплохо бы добавить метод **finish()**, чтобы закрыть вторую активность за ненадобностью. Если переход происходит через кнопку Назад, то это делать не обязательно.

Если активность была закрыта пользователем при нажатии аппаратной кнопки возврата или если метод **finish()** был вызван раньше, чем метод **setResult()**, результирующий код установится в **RESULT\_CANCELED**, а возвращенное намерение покажет значение **null**.

Возвращаемся на первый экран. Первый экран ожидает ответа от второго экрана, поэтому нужно добавить в код метод **onActivityResult()**.

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

TextView infoTextView = (TextView) findViewById(R.id.textViewInfo);

if (requestCode == CHOOSE\_THIEF) {

if (resultCode == RESULT\_OK) {

String thiefname = data.getStringExtra(ChooseActivity.THIEF);

infoTextView.setText(thiefname);

}else {

infoTextView.setText(""); // стираем текст

}

}

}

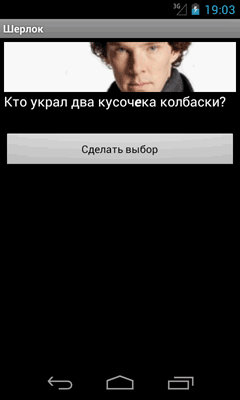
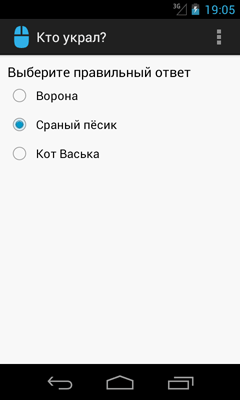
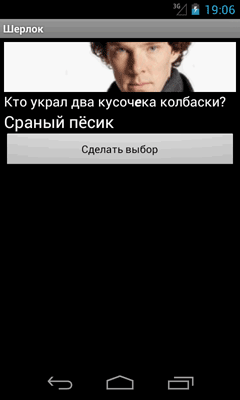
Метод ожидает входящие данные с кодом **CHOOSE\_THIEF**, и если такие данные поступят, то извлекает значение из ключа **ChooseActivity.THIEF** с помощью метода **getStringExtra**. Полученное значение мы выводим в **TextView** (переменная **infoTextView**). Если мы вернулись на экран через кнопку Back, то просто стираем текст.

При закрытии дочерней активности внутри родительского компонента срабатывает обработчик **onActivityResult()**. Обработчик **onActivityResult()** принимает несколько параметров.

* Код запроса. Код, который использовался для запуска активности, возвращающей результат
* Результирующий код. Код результата, устанавливаемый дочерней активностью и указывающий, как завершилась её работа. Это может быть любое целочисленное значение, но, как правило, либо **Activity.RESULT\_OK**, либо **Activity.RESULT\_CANCELED**
* Данные. Намерение, используемое для упаковки возвращаемых данных. В зависимости от назначения дочерней активности оно может включать путь URI, представляющий выбранную часть содержимого. В качестве альтернативы (или дополнения) дочерняя активность может возвращать информацию в виде простых значений, упакованных в параметр намерения **extras**

Если работа дочерней активности завершилась непредвиденно или если перед её закрытием не был указан код результата, этот параметр станет равен **Activity.RESULT\_CANCELED**.

Запускаем проект, нажимаем на кнопку и переходим на второй экран. Там выбираем один из вариантов. Если выбрать ворону, то экран закроется и имя преступника отобразится на первом экране. Если выбрать пёсика, то отобразится его имя.

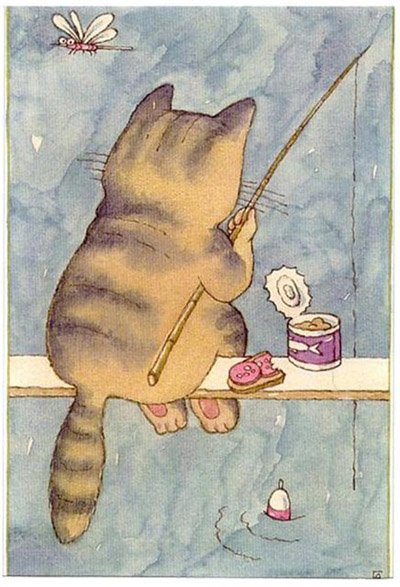
  

Между прочим, если выбрать котика, то его имя не отобразится! Проверьте и убедитесь сами. Вы спросите почему? Элементарно, Ватсон! Преступник не учёл одной важной детали. В ресторане велось наблюдение с видеокамер, и запись показала, кто на самом деле украл колбаску и подставил кота. Васька, держись!



P.S. Если поначалу что-то показалось непонятным, то с практикой многое прояснится. Передача данных между экранами часто встречается в приложениях и вы ещё не раз изучите пример.

P.P.S. Лучшая рыба - колбаса. Зная эту слабость, нетрудно было подставить кота.



## Используем фильтры

В статье я показывал распространённый способ перехода на другую активность, когда в методе **startActivity()** указывается текущий класс и класс для перехода. Кстати, класс активности не обязательно должен быть частью вашего приложения. Если вы знаете имя класса из другого приложения, то можете перейти и на него. Но можно перейти в другую активность другим способом.

На практике встречается реже, но может пригодиться. Допустим, у вас уже есть вторая активность. В манифесте добавим к ней специальный фильтр:

<activity

android:name=".SecondActivity"

android:label="@string/title\_activity\_second" >

**<intent-filter >**

**<action android:name="ru.alexanderklimov.testapplication.SecondActivity" />**

**<category android:name="android.intent.category.DEFAULT" />**

**</intent-filter>**

</activity>

И запускаем вторую активность через щелчок кнопки таким способом.

public void onClick(View view) {

startActivity(new Intent("ru.alexanderklimov.testapplication.SecondActivity"));

}

Заменим длинную строку на константу.

public static final String ACTION\_SECOND\_ACTIVITY = "ru.alexanderklimov.testapplication.SecondActivity";

public void onClick(View view) {

startActivity(new Intent(ACTION\_SECOND\_ACTIVITY));

}

Итак, что мы сделали. Для второй активности мы прописали фильтр и указали имя для **action** в атрибуте **android:name**. Для удобства я просто поместил в него полное имя активности с названием пакета. Конструктор класса **Intent** имеет несколько перегруженных версий. В одной из версий можно указать строку для действия. Мы указали своё созданное действие, которое прописано у второй активности. Система во время работы просматривает манифесты всех установленных приложений. При поиске соответствия система находит наш фильтр и запускает нужную активность.

По такому же принципу можно запустить другие активности. Посмотрите на пример [Открываем окно настроек для автономного режима](http://developer.alexanderklimov.ru/android/theory/airplanemode.php#settings). Если вы скопируете пример к себе и посмотрите на документацию по **android.provider.Settings.ACTION\_AIRPLANE\_MODE\_SETTINGS**, то увидите, что этому коду соответствует строковая константа **public static final java.lang.String ACTION\_AIRPLANE\_MODE\_SETTINGS = "android.settings.AIRPLANE\_MODE\_SETTINGS"**. Сравните с нашим кодом. Вы можете предположить, что у активности настроек для автономного режима в фильтре прописана эта строка.

Имя категории фильтра **android.intent.category.DEFAULT** говорит системе, что следует выполнить действие по умолчанию, а именно, запустить активность. Существует и другие имена, которые пока нас не интересуют.

А теперь вопрос на засыпку. Что произойдёт, если создать ещё одну активность и указать такой же фильтр, как у второй активности? А давайте проверим. Создайте у себя третью активность и скопируйте блок с фильтром от второй активности в него.

<activity

android:name=".ThirdActivity"

android:label="@string/title\_activity\_third" >

<intent-filter>

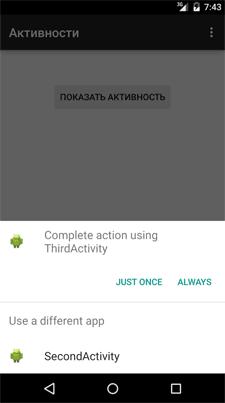
<action android:name="ru.alexanderklimov.testapplication.SecondActivity" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

Щёлкаем по кнопке в первой активности. Система попросит выбрать нужный вариант.



Если вы выберите пункт **ALWAYS**, то в следующий раз выбирать не придётся. Чтобы сбросить выбор, зайдите в свойства приложения в Настройках и найдите кнопку **Clear defaults**.