**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 и т.д. Воспринимайте экран активности как веб-страницу с ссылкой на другую страницу.

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.

* Создайте новый проект с одной кнопкой на экране.
* Далее создадим новую форму для отображения полезной информации.

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 по 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

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.

## Обработки щелчка кнопки

Google рекомендует использовать данный способ как удобный, требующий меньше кода и понятный для чтения. Добавьте в проект новую кнопку и разместите под первой кнопкой. Подправьте несколько атрибутов, например, так.

<Button

android:id="@+id/button\_сounter"

... другие атрибуты

android:text="Считаем ворон" />

Я поменял для кнопки текст и идентификатор, а также некоторые параметры, отвечающие за положение кнопки на экране. Переключитесь в графический режим и посмотрите, как выглядит кнопка. Обратите внимание, что атрибут **android:onClick** у второй кнопки мы уже не используем.

Снова открываем файл **MainActivity.java** и пишем следующий код сразу после объявления класса и до метода **onCreate()**:

private Button mCrowsCounterButton;

Во время набора студия будет пытаться угадать, что вы хотите ввести. Активно используйте подсказки. Например, уже при первом вводе символа **B** студия предложит несколько вариантов на эту букву. Если нужное слово находится первым в списке, то нажимайте клавишу Enter, иначе выберите нужное слово из списка и потом уже нажимайте Enter. Это удобно при наборе длинных имён классов и переменных.

Google разработал целое руководство по наименованию переменных. Например, закрытая переменная на уровне класса должна начинаться с символа **m** (member), а далее идёт понятное название с заглавной буквы. Давайте попробуем придерживаться этого стиля.

После строчки **setContentView(R.layout.activity\_main);** пишем:

mCrowsCounterButton = findViewById(R.id.buttonCrowsCounter);

Ещё раз напоминаю, не копируйте текст со страницы, а вводите текст вручную. Так вы быстрее освоите программирование, кроме того научитесь пользоваться системой подсказок и уменьшите количество опечаток.

Переходим к самому важному - обработчику щелчка кнопки. Нам понадобится дополнительная переменная-счётчик **mCount**, которая будет содержать число подсчитанных ворон (её необходимо разместить выше метода **onCreate()** рядом с переменной **mCrowsCounterButton**).

private int mCount = 0;

Мы создали переменную и сразу её инициализировали.

Теперь пишем обработчик щелчка кнопки, активно используя всплывающие подсказки для быстрого набора. Код добавляется в методе **onCreate()** после других строк, написанных ранее в этом методе.

Попробую описать, как использовать автодополнение. Сначала начинаем вводим первые символы слова **mCrowsCounterButton** (можно маленькими буквами) и нажимаем Enter, если видим, что нужная подсказка появилась. После этого слова ставим точку и должны появиться опять подсказки, которые относятся к данной переменной. Начинаем вводим первые буквы слова **setOnClickListener**.

Здесь тоже проблем обычно не возникает. На данный момент у нас получилась строка **mCrowsCounterButton.setOnClickListener();**.

Ставим курсор внутри круглых скобок и набираем **new OnClickListener**. Здесь важно набрать символ **O** в верхнем регистре. Тогда у вас появится нужная подсказка типа **View.OnClickListener{...} (android.view.View)**. Нажимаем Enter и получаем нужную заготовку, внутри которой вставляем код:

mCrowsCounterButton.setOnClickListener(new View.OnClickListener() {

 @Override

 public void onClick(View v) {

 helloTextView.setText("Я насчитал " + ++mCount + " ворон");

 }

});

Ранее мы объявили переменную **helloTextView** внутри метода **onClick()**, из которого она недоступна в других методах. Поэтому поступим с ней так же, как с кнопкой - объявим текстовую метку на уровне класса и инициализируем её в методе **onCreate()**. Сделайте это самостоятельно, создав переменную с именем **mInfoTextView**.

Запускаем приложение и начинаем щёлкать по кнопке. При каждом щелчке счётчик **mCount** будет увеличиваться на единицу и эта информация будет отображаться на экране.

Нет смысла дублировать код для одной текстовой метки. Поэтому для первой кнопки, которая здоровалась, код можно сократить.

public void onClick(View view) {

 mInfoTextView.setText("Hello Kitty!");

}

Мы познакомились с новым способом обработки щелчка кнопки. Какой способ вы предпочтёте - зависит от ваших привычек и вкуса. Более того, есть ещё один способ, о котором можно прочитать в отдельной статье о кнопках.

Вдобавок, если подключить к проекту [Java 8](http://developer.alexanderklimov.ru/android/java/java8.php), то вместо кода с подключением слушателя, можно использовать лямбда-выражение. Студия сама предложит заменить код.

mCrowsCounterButton.setOnClickListener(v ->

 mInfoTextView.setText("Я насчитал " + ++mCount + " ворон"));



Обработка щелчка кнопки