Старт другой активности Android Studio

BY [МАРИНА](https://maryone.ru/author/mary/) · 03.01.2018

После того как у нас есть один готовый экран, время узнать больше. Мы добавим некоторый код в нашу Activity (активность), который позволить запустить новую активность. Новая Activity отобразит сообщение в ответ на нажатие пользователем кнопки Отправить.

В этом уроке будет следующее:

* Поработаем над кнопкой «Отправить» (Send)
* Как создать Intent (намерение)
* Создадим вторую Activity (активность)
* Добавить TextView (текстовое поле)
* Отобразим сообщение
* Добавим навигацию

Поработаем над кнопкой «Отправить» (Send)

У нас есть кнопка «Отправить», но при ее нажатии ничего не происходит. Время это исправить. Добавим следующий код (он отмечен жирным):

public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

**/\*\* Вызывается когда пользователь нажимает кнопку "Отправить" \*/**

**public void sendMessage(View view) {**

**// Здесь можно описать действия, которые будут происходить при нажатии кнопки**

**}**

}

В методе sendMessage в качестве аргумента используется экземпляр view класса View. Он подчеркнут красным, как ошибка. Чтобы ее исправить необходимо импортировать класс. Для этого курсор мыши ставим на View, нажимаем сочетание клавиш Alt+Enter — выбираем Import Class.

Теперь присвоим указанный код нашей кнопке. Возвращаемся в  **activity\_main.xml,**переходим в режим отображения макета. Находим нашу кнопку, выбираем её щелчком мыши и в появившихся атрибутах справа ищем атрибут onClick. В поле напротив из выпадающего списка выбираем созданный нами метод sendMessage.

Чтобы метод распознавался в качестве возможного метода onClick необходимо выполнить три обязательных условия:

1. метод должен быть публичным public
2. он не должен ничего возвращать, и быть void
3. в качестве параметра в скобках должно быть только View

Как создать Intent (намерение)

Intent — это объект, который обеспечивает привязку времени выполнения между отдельными компонентами, такими как дву Activity. Intent представляет собой «намерение сделать что-то». Вы можете использовать намерения для самых разных задач, но сейчас мы рассмотрим как перейти из одной Activity на другую.

В MainActivity.java добавьте константу EXTRA\_MESSAGE и код sendMessage () исправьте, как показано ниже (отмечено жирным красным):

public class MainActivity extends AppCompatActivity {

**public static final String EXTRA\_MESSAGE = "com.example.myfirstapp.MESSAGE";**

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    /\*\* Called when the user taps the Send button \*/

    public void sendMessage(View view) {

**Intent intent = new Intent(this, DisplayMessageActivity.class);**

**EditText editText = (EditText) findViewById(R.id.editText);**

**String message = editText.getText().toString();**

**intent.putExtra(EXTRA\_MESSAGE, message);**

**startActivity(intent);**

    }

}

Снова столкнемся с ошибками, которые будут подчеркнуты красным — например класс Intent. Его так же импортируем (нажимаем сочетание клавиш Alt+Enter — выбираем Import Class).

Проверьте, должны быть импортированы следующие элементы:

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

Если у вас не так, проверьте еще раз.

Сейчас не обращаем внимания только на ошибку, связанную с DisplayMessageActivity, так как саму новую Activity мы создадим немного позднее.

Сейчас остановимся на том, что мы написали в  sendMessage ().

Конструктор Intent принимает два параметра — контекст (потому что Activity является подклассом Context), класс компонента приложения, которому система должна предоставить намерение (то есть Activity, которая будет запущена).

Метод putExtra () добавляет значение EditText к намерению. Intent может переносить типы данных в виде пар ключ-значение, называемых дополнительными. Ваш ключ является общедоступной константой EXTRA\_MESSAGE. И следующая Activity по ключу может получить переданное значение.  Хорошая практика заключается в определении ключей для дополнительных намерений, использующих имя пакета приложения в качестве префикса. Это гарантирует, что ключи уникальны, если ваше приложение взаимодействует с другими приложениями.

Итак, теперь метод startActivity () запускает экземпляр DisplayMessageActivity, указанный в Intent. Теперь вам нужно создать этот класс.

Создадим вторую Activity (активность)

1. В левой части экрана (Project) щелкаем правой кнопкой мыши в структуре нашего приложения и выбираем: **New > Activity > Empty Activity**.
2. В окне **Configure Activity**введите название новой Activity **DisplayMessageActivity**и нажмите Готово (Finish), больше ничего не меняя.

Android Studio автоматически выполняет три действия:

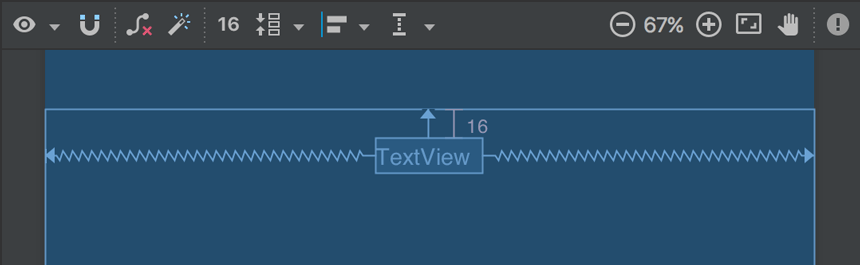
* Создает файл DisplayMessageActivity.java.
* Создает соответствующий файл макета activity\_display\_message.xml.
* Добавляет необходимый элемент в AndroidManifest.xml.

Если вы запустите приложение и нажмите кнопку в первом действии, запустится второе Activity, но оно будет пустым. Это связано с тем, что  макет всё еще пустой.

Добавить TextView (текстовое поле)

Новое Activity включает пустой файл макета, поэтому теперь добавим текстовое поле, в котором появится сообщение.

1. Откройте файл app> res> layout> activity\_display\_message.xml.
2. Нажмите «Включить автосоединение»(Turn On Autoconnec) на панели инструментов (он должен быть включен, как показано на рисунке 1).
3. В окне «Палитра»(Palette) нажмите «Текст»(**Text**), а затем перетащите **TextView** в макет — поместите его рядом с верхней частью макета, рядом с центром, чтобы он привязывался к вертикальной линии. Autoconnect добавляет левые и правые ограничения, чтобы поместить представление в горизонтальный центр.
4. Создайте еще одно ограничение в верхней части текстового вида в верхней части макета, чтобы оно отображалось, как показано на рисунке 1.



Вы можете настроить размер текста и его цвет как вам нравится, изменив атрибуты текстового поля textSize и textColor.

Отобразим сообщение

Теперь изменим вторую Activity так, чтобы отобразить сообщение, которое было передано первым Activity.

1. В DisplayMessageActivity.java добавьте следующий код к методу onCreate ():

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity\_display\_message);

**// Get the Intent that started this activity and extract the string**

**Intent intent = getIntent();**

**String message = intent.getStringExtra(MainActivity.EXTRA\_MESSAGE);**

**// Capture the layout's TextView and set the string as its text**

**TextView textView = findViewById(R.id.textView);**

**textView.setText(message);**

}

2. Нажмите Alt + Enter (или Option + Return на Mac), чтобы импортировать отсутствующие классы. Ваш импорт должен быть следующим:

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.widget.TextView;

Добавим навигацию

Каждый экран вашего приложения, который не является основным входом (все экраны, которые не являются «домашним» экраном), должен обеспечивать навигацию, чтобы пользователь мог вернуться на экран логического родителя в иерархии приложений, нажав кнопку «Вверх» в панели приложений.

Все, что вам нужно сделать, это объявить, какая активность является логическим родителем в файле AndroidManifest.xml. Поэтому откройте файл в app> manifestests> AndroidManifest.xml, найдите тег для DisplayMessageActivity и замените его следующим:

<activity< span=""> android:name=".DisplayMessageActivity"

          android:parentActivityName=".MainActivity" >

    <meta-data< span="">

        android:name="android.support.PARENT\_ACTIVITY"

        android:value=".MainActivity" />

</meta-data<></activity<>

Система Android теперь автоматически добавляет кнопку «Вверх» в панели приложений.

Запускаем приложение!

Скорее оценим, как преобразилось наше приложение.

Запустите приложение снова, нажав кнопку «Применить изменения»(Apply Changes) на панели инструментов. Когда он откроется, введите сообщение в текстовое поле и нажмите «Отправить», чтобы увидеть, как сообщение появляется во втором действии.



Вот и все, вы создали свое первое приложение для Android!