Создание первого приложения

Практическая работа 1

Методические указания

Создание первого приложения

Это занятие поможет вам создать первое приложение для Android. Вы научитесь создавать проекты и запускать ваше приложение в режиме отладки. Вы так же узнаете об основах разработки приложения для Android, включая создание простого пользовательского интерфейса и обработку ввода информации пользователем.

Установка среды разработки

Перед тем как начать изучение, вам необходимо установить инструменты для разработки:

- Скачать Android Studio. (https://developer.android.google.cn/studio?hl=id)
- Скачать последнюю версию инструментов SDK, используя SDK Manager.(https://www.techspot.com/downloads/5425-android-sdk.html)

Примечание: Хотя большинство практик предполагает использование Android Studio, в некоторых случаях будем приводиться примеры работы с инструментами SDK в командной строке.

Данное занятие содержит пошаговые инструкции по созданию небольшого приложения, которые позволят вам узнать об основных понятиях, встречающихся при разработке Android приложений.

Очень важно пройти занятие шаг за шагом от начала и до конца.

Создание проекта

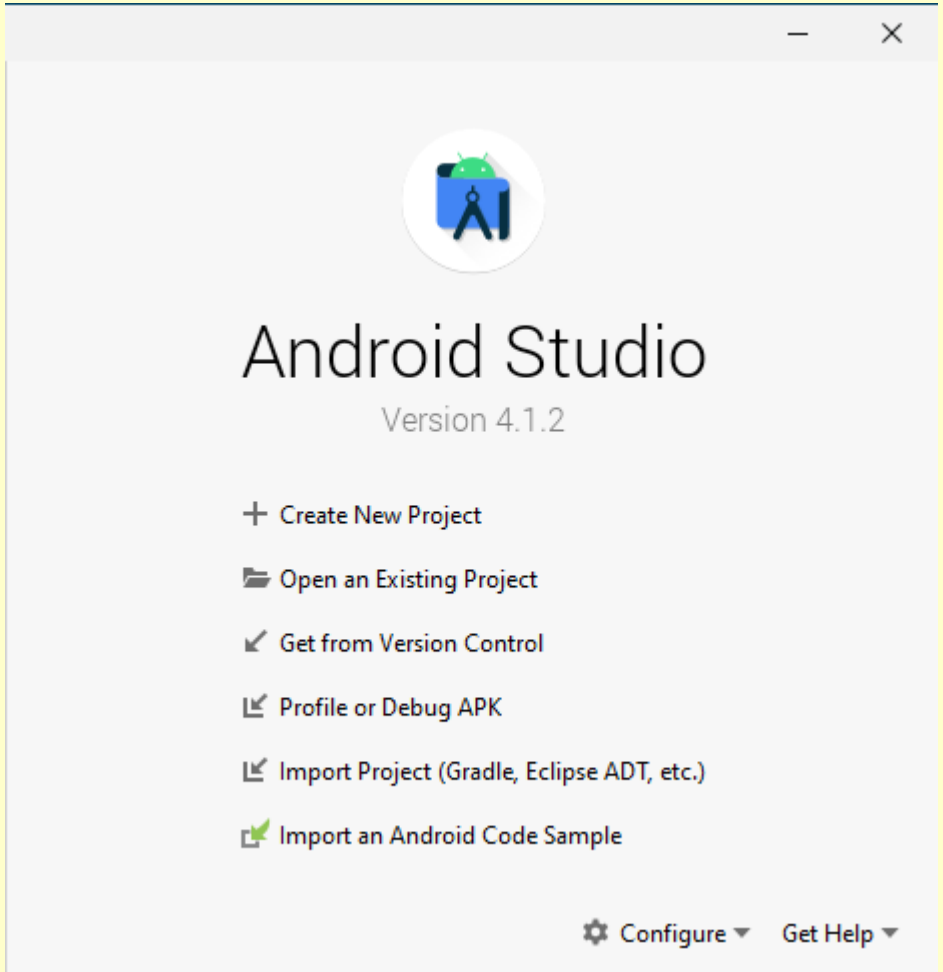
Проект Android включает в себя все файлы, которые содержат исходный код приложения.

В данном материале рассказывается о двух способах создания нового проекта: в Android Studio либо в командной строке, используя инструментарий SDK.

Создание проекта в Android Studio

1.Создайте новый проект в Android Studio:

- Щелкните New Project на экране приветствия.
- Если открыт другой проект, щелкните меню File и выберите New Project.



В стартовом окне нажмите Configure(Выберите AVD Manager,выберите устройство, на котором можно запускать приложение)

2. Заполните поля в окне Configure your new project. Будет проще, если вы введете значения, как указано на рисунке 1.

- **Name** - это название приложения, которое увидят пользователи. Для текущего проекта напишите “My First App”.
- **Minimum SDK -** это минимальное значение API.
- **Language** - язык программирования
- **Package name** - это полное наименование пакета проекта (согласно правилам именования пакетов в языке Java). Название пакета должно быть уникальным и не может совпадать с названиями других пакетов, установленных на устройстве. Вы можете использовать любое название пакета, независимо от названия приложения или пространства имен.
- **Save location** - это директория на вашем компьютере в которой хранятся все файлы проекта.

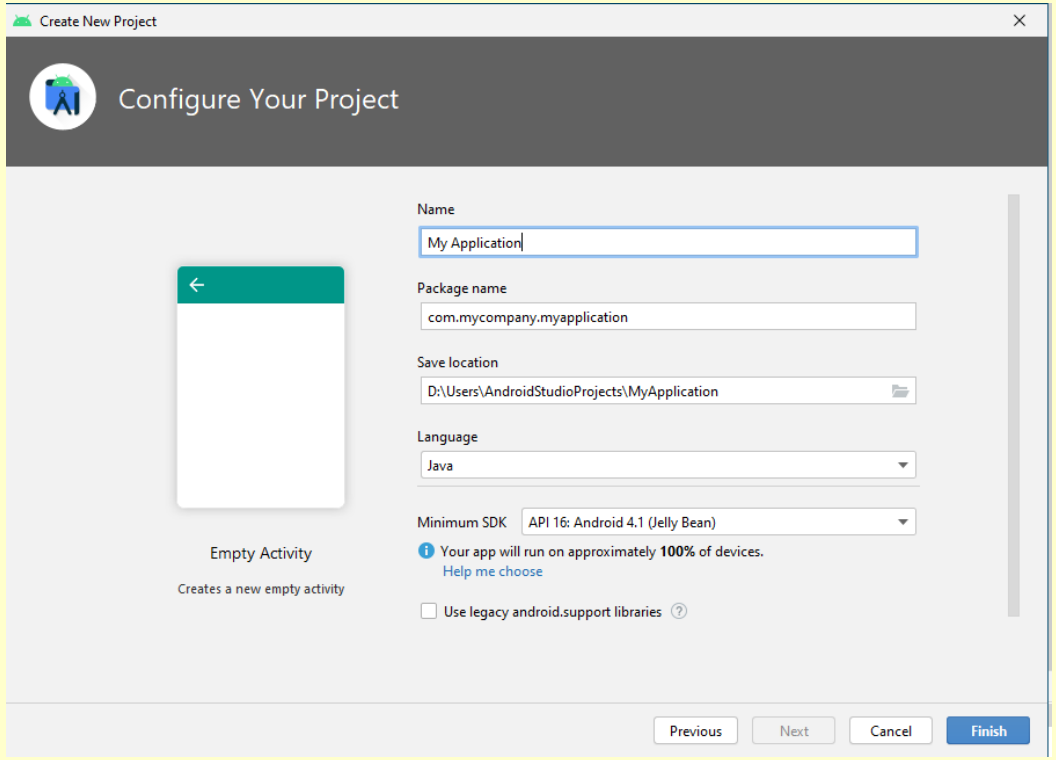


Рисунок 1. Создание нового проекта в Android Studio

Выберите API 16: Android 4.1 (Jelly Bean) в поле Minimum SDK.

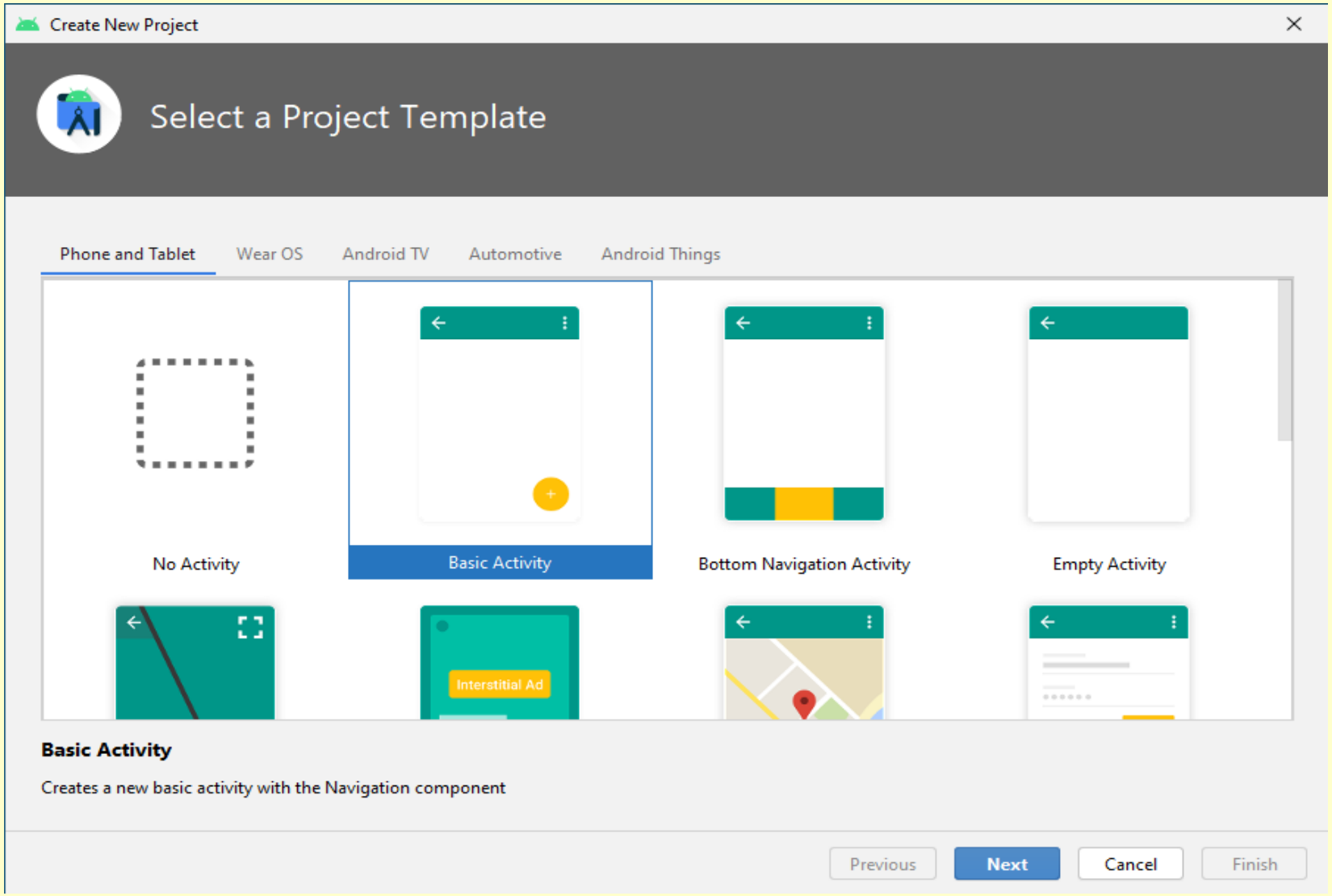
Минимально необходимый SDK – это младшая версия Android, на котором ваше приложение сможет работать.

Для указания версии используются уровни API.

Для поддержки как можно большего числа устройств, вы должны выбрать наименьшую версию API, при которой ваше приложение выполняет свой основной набор функций.

Если какие-либо некритичные функции вашего приложения работают только на новых версиях Android, вы можете включать их только при запуске на устройствах с определенной версией

Выберите закладку Phone and Tablet, а за тем в ней, бланк активности **Empty Activity** (пустое явление) и нажмите Next.



Что такое Явление (Activity)

Явление – это одна из отличительных особенностей Android фреймворка. Явления предоставляют пользователю доступ к вашему приложению. Приложение может содержать несколько явлений. Обычно приложение содержит главное явление, которое отображается при запуске и прочие явления, для выполнения разных задач в рамка приложения, например для просмотра документов. Можно воспринимать явления как отдельный экран приложения, аналог окна в других операционных системах, включающий в себя элементы интерфейса и программный код.

Будет создан новый проект, содержащий некоторые стандартные файлы. Рассмотрим наиболее важные из них:

app/src/main/res/layout/activity_my.xml

Это XML файл разметки явления мы добавили при создании проекта в Android Studio. После создания нового проекта, Android Studio показывает этот файл в текстовом виде и в режиме предпросмотра экрана устройства. Файл по умолчанию содержит элемент TextView, который отображает строку “Hello world!”

app/src/main/java/com.mycompany.myfirstapp/MyActivity.java

Закладка с этим файлом появляется в Android Studio сразу после создания проекта. Файл содержит описание класса созданного явления. Если вы скомпилируете и запустите приложение, класс Activity загрузит файл разметки и отобразит надпись “Hello, world!”

app/src/res/AndroidManifest.xml

Файл манифеста описывает основные характеристики приложения и всех его компонентов. Мы вернемся к этому файлу позже, когда будем добавлять компоненты в приложение.

app/build.gradle

Android Studio использует Gradle для компиляции и сборки приложений. Существуют файлы build.gradle как для каждого модуля, так и для всего приложения в целом. Обычно вы будете иметь дело с файлами для модулей, в данном примере для модуля app. В файле указываются зависимости приложения, а так же стандартные настройки:

- compiledSdkVersion – версия платформы, на которой будет производиться компиляция приложения. По умолчанию это последняя версия Android, установленная в вашем SDK. (Вы должны использовать Android 4.1 или выше. Установите нужную версию, используя SDK Manager). Вы все еще можете создавать приложения с поддержкой старых версий, однако выбор последней версии для сборки приложения позволяет использовать новые возможности и оптимизировать работу приложения для свежих устройств.
- applicationId – полное наименование пакета вашего приложения, которое вы указали при создании проекта.
- minSdkVersion – минимально необходимая версия андроид, которую вы указали при создании проекта. Это младшая версия андроид, которую поддерживает ваше приложение.
- targetSdkVersion – указывает на самую большую версию андроид, на которой вы проверяли работу вашего приложения. Рекомендуем тестировать приложение при выходе каждой новой версии андроид и увеличивать значение targetSdkVersion. Таким образом вам будут доступны новые возможности платформы.

Также отметим содержимое директории /res, которая содержит ресурсы приложения:

drawable-hdpi/

Содержит нарисованные объекты, такие как растровые изображения, для экранов высокого разрешения (hdpi). В других директориях drawable содержатся изображения для экранов с разным разрешением. К примеру в ней находится файл ic_launcher.png, содержащий иконку приложения.

layout/

Содержит файлы разметки пользовательского интерфейса. К примеру файл activity_my.xml, о котором говорилось выше, содержит разметку для класса MyActivity.

values/

Содержит XML файлы, включающие в себя такие ресурсы, как строки и определение цветов. К примеру в файле string.xml хранится строка “Hello world!”, отображаемая при запуске приложения.

Создание проекта с использованием командной строки

Если вы не используете среду разработки Android Studio, вы можете создать проект используя командную строку:

1. Перейдите в директорию tools/ вашего Android SDK.

2. Выполните команду:

android list targets

Данная команда выводит список доступных в вашем SDK платформ. Найдите платформу, на которой вы хотите выполнять компиляцию вашего приложения и запомните targetID. Мы рекомендуем выбирать максимально доступную версию. Вы все еще можете создавать приложения с поддержкой старых версий, однако выбор последней версии для сборки приложения позволяет использовать новые возможности и оптимизировать работу приложения для свежих устройств.

Если список доступных платформ пуст, вам необходимо их установить, используя Android SDK Manager.

3. Выполните команду:

android create project --target <target-id> --name MyFirstApp \

--path <path-to-workspace>/MyFirstApp --activity MyActivity \

--package com.example.myfirstapp

Где <target-id> замените на идентификатор из списка доступных платформ, а вместо <path-to-workspace> укажите директорию для хранения файлов проекта.

Совет: добавьте директории platform-tools/ и tools/ в переменную окружения PATH.

Запуск вашего приложения

- Если вы следовали инструкциям предудыщего материала, то сейчас у вас есть все необходимое для немедленного запуска приложения.
- Как вы будете запускать приложение зависит от двух вещей: имеется ли у вас реальное Android устройство и используете ли вы Android Studio.

Запуск на реальном устройстве

Если у вас есть устройство, работающее под Android, следуйте инструкциям ниже для установки и запуска приложения.

Подготовка вашего устройства

1. Подключите ваше устройство к компьютеру с помощью USB кабеля. Если вы используете операционную систему Windows, может понадобиться установка USB драйвера для вашего устройства. Подробнее об установке драйверов читайте в разделе OEM USB драйвера.

2. Включите на устройстве режим USB отладки.

- На большинстве устройств, работающих под управлением Android 3.2 и старше данная опция находится в меню Настройки > Приложения > Разработка
- В андроид 4.0 и новее опция находится в меню Настройка > Разработка.

Примечание: В Android 4.2 и новее, пункт меню Разработка по умолчанию скрыт. Чтобы отобразить его в меню, нажмите Настройки > О телефоне и щелкните по пункту Номер сборки семь раз. После этого вернитесь в предыдущее меню и найдите пункт Разработка.

Запуск приложения из Android Studio

1. Выберите один из файлов вашего проекта и нажмите кнопку Run на панели инструментов.

2. В появившемся окне Choose Device (выбор устройства), активируйте пункт Choose a running device (выбрать подключенное устройство) и выберите ваш телефон. Нажмите ОК.

Android Studio установит приложение на подключенное устройство и запустит его.

Запуск приложения из командной строки

1. Перейдите в корневую директорию вашего проекта и выполните команду

ant debug

2. Убедитесь, что директория platform-tools/ добавлена в переменную окружения PATH и выполните команду:

adb install bin/MyFirstApp-debug.apk

Найдите на вашем устройстве приложение MyFirstApp и запустите его.

Как видите, собрать и запустить приложение на вашем устройстве очень просто!

Запуск приложения в эмуляторе

1. Запустить менеджер виртуальных устройств можно двумя способами:

В Android Studio выберите Tools > Android > AVD Manager или щелкните по иконке менеджера на панели инструментов.

В командной строке перейдите в директорию <sdk>/tools/ и выполните команду:

android avd

Примечание: Внешний вид менеджера виртуальных устройств, запущенного из командной строки отличается от менеджера в Android Studio, поэтому действия, показанные ниже могут отличаться.

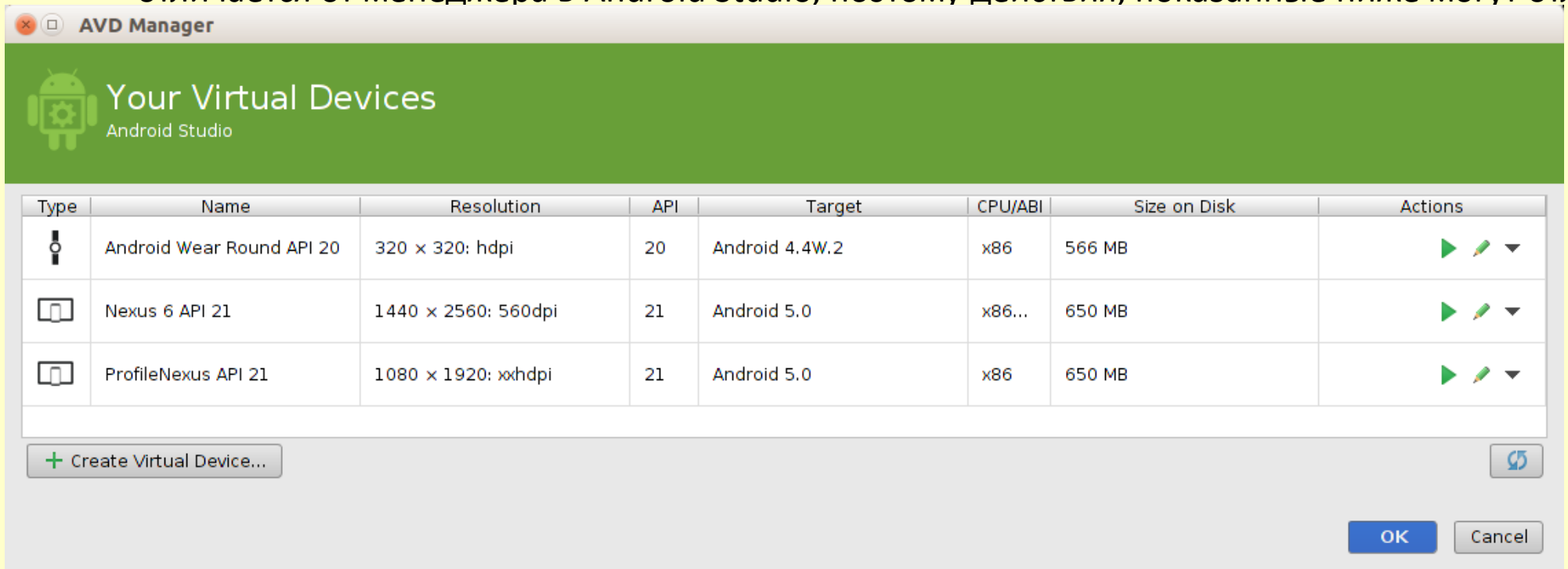


Рисунок 2. Менеджер виртуальных устройств

2. В окне менеджера устройств щелкните Create Virtual Device.

3. В следующем окне выберите конфигурацию подходящего устройства, например Nexus 6 и нажмите Next.

4. Выберите желаемую версию AVD и нажмите Next.

5. Проверьте правильность заполнения конфигурации и нажмите Finish.

Информацию об использовании AVD рассмотрим позднее

Запуск приложения из Android Studio

1. Выберите один из файлов вашего проекта и нажмите кнопку Run на панели инструментов.

2. В появившемся окне Choose Device (выбор устройства), активируйте пункт Launch emulator (выбрать эмулятор)

3. В списке виртуальных устройств выберите созданный вами эмулятор и нажмите ОК.

Запуск эмулятора может занять несколько минут. После разблокировки экрана вы увидите запущенное приложение.

Запуск приложения из командной строки

1. Перейдите в корневую директорию вашего проекта и выполните команду

ant debug

2. Убедитесь, что директория platform-tools/ добавлена в переменную окружения PATH и выполните команду:

adb install bin/MyFirstApp-debug.apk

Найдите в эмуляторе приложение MyFirstApp и запустите его.

Создание простого интерфейса пользователя

В данном материале мы создадим XML разметку, включающую текстовое поле и кнопку. В следующем материале мы научимся обрабатывать нажатие на кнопку и передавать данные из текстового поля в другое явление.

Графический интерфейс пользователя в Android строится на основе дерева объектов типа View и ViewGroup. Объекты типа View – это обычные виджеты, такие как кнопки и поля ввода. Объекты типа ViewGroup это невидимые контейнеры, которые управляют расположением вложенных элементов, например в виде таблицы или вертикального списка.

Android предоставляет набор XML элементов, соответствующих наследникам View и ViewGroup, чтобы вы могли описывать интерфейс пользователя с помощью XML.

Объекты разметки (Layouts) являются наследниками класса ViewGroup. В данном материале мы будем работать с линейной разметкой LinearLayout.

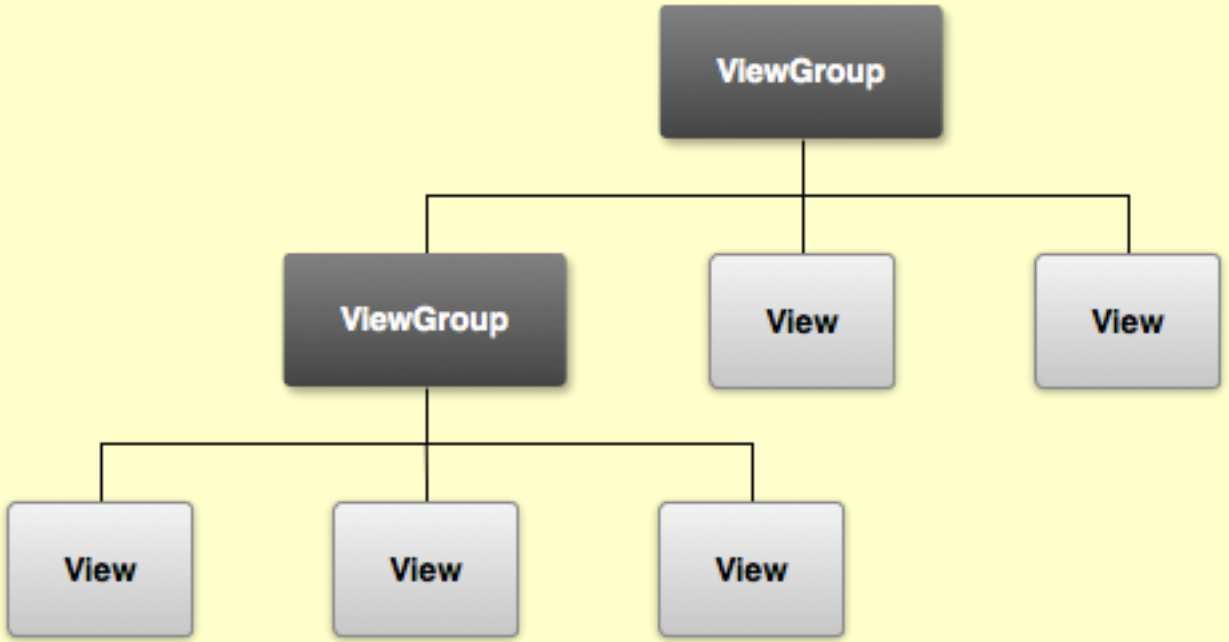


Рисунок 3 .ViewGroup позволяет строить дерево объектов разметки.

Различная разметка

Создание разметки в XML файлах предпочтительнее, чем в исходном коде по нескольким причинам, но главным образом из-за необходимости создания различных файлов разметки для устройств с различными размерами экрана.

К примеру, вы можете создать две версии разметки и заставить систему отображать одну из них на маленьких экранах, а другую на больших.

Создание линейной разметки (Linear Layout)

1. В Android Studio откройте файл res/layout/activity_my.xml

Шаблон BlankActivity, который вы выбрали при создании проекта включает в себя файл activity_my.xml, который содержит корневой элемент RelativeLayout с вложенным виджетом TextView.

3. В окне предпросмотра щелкните по иконке Hide , чтобы скрыть окно.

В Android Studio при открытии файла разметки по умолчанию открывается окно предпросмотра, которое содержит WYSIWYG редактор для создания интерфейса пользователя. Однако в данном материале мы будем работать напрямую с XML файлом.

3. Удалите элемент <TextView>.

4. Измените элемент <RelativeLayout> на <LinearLayout>.

5. Добавьте атрибут android:orientation и установите для него значение "horizontal".

6. Удалите атрибуты android:padding и tools:context.

В результате разметка должна выглядеть так:

res/layout/activity_my.xml

```
<LinearLayout xmlns:android="schemas.android.com/apk/res/android"
    xmlns:tools="schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
</LinearLayout>
```

LinearLayout является наследником класса ViewGroup и размещает вложенные элементы в вертикальном или горизонтальном порядке, в зависимости от значения атрибута android:orientation. Вложенные в LinearLayout элементы отображается на экране в том порядке, в котором они описаны в XML.

Все элементы разметки должны иметь атрибуты android:layout_width и android:layout_height, которые указывают на их размер.

Поскольку LinearLayout корневой элемент разметки, он должен быть растянут на весь экран. Для этого укажем значение атрибутов android:layout_width и android:layout_height равное "match_parent". Это означает, что ширина и высота элемента должна соответствовать ширине и высоте родительского элемента.

Добавление текстового поля

Как и для каждого объекта типа View, необходимо указать некоторые XML атрибуты, характерные для элемента EditText.

1. В файле activity_my.xml, создайте внутри <LinearLayout> элемент <EditText> и укажите для него атрибут android:id со значением @+id/edit_message.

2. Создайте атрибуты layout_width и layout_height со значением "wrap_content".

3. Создайте атрибут hint и укажите в качестве значения строковый объект с названием edit_message.

В результате элемент <EditText> должен выглядеть следующим образом:

res/layout/activity_my.xml

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

Атрибуты элемента <EditText> означают следующее:

android:id

Этот атрибут задает уникальный идентификатор элемента, по которому вы можете ссылаться на объект из программного кода и работать с ним. (Как это делается вы узнаете в следующем уроке). Значок “@” требуется указывать, если вы ссылаетесь на какой-либо объект ресурса из XML. После значка идет тип ресурса (в данном примере id), затем имя ресурса после слэша (edit_message).

Значок “+” перед типом ресурса необходим только при первоначальном указании идентификатора. При компиляции приложения, SDK использует указанный идентификатор для создания переменной в файле gen/R.java. Эта переменная будет указывать на элемент EditText. После этого в указании значка “+” нет необходимости. Другими словами, значок “+” необходимо указывать только при создании нового идентификатора. Нет необходимости добавлять “+” при работе с существующими ресурсами, такими как строки или разметка.

android:layout_width и android:layout_height

Вместо указания конкретных размеров элемента, мы указали "wrap_content", что позволяет элементу менять размер в зависимости от содержимого. Если мы зададим значение "match_parent", то элемент EditText заполнит весь экран, поскольку его ширина и высота будут равны родительскому LinearLayout.

android:hint

Этот атрибут задает подсказку, которая отображается в текстовом поле, если оно не заполнено.

Вместо жестко заданной строки мы указали ссылку на строковый ресурс "@string/edit_message", который находится в другом файле. Поскольку это ссылка на конкретный ресурс, нет необходимости добавлять значок “+”. Однако поскольку вы еще не создали данный строковый ресурс, компилятор выдаст ошибку. Мы добавим ресурс немного позже.

Примечание: Указанный строковый ресурс имеет такое же имя, как и идентификатор элемента. Однако при ссылке на ресурс всегда учитывается его тип (например id или string), поэтому использование одинаковых имен не вызывает проблем.

Объекты ресурсов

Объект ресурса имеет уникальное целочисленное значение, связанное с ресурсами приложения, такими как растровые изображения, файлы разметки или строки.

Каждый ресурс соответствует объекту ресурса, объявленному в файле gen/R.java.

Вы можете использовать имя объекта из класса R для ссылки на ресурсы из кода, например если вам нужно программно изменить атрибут android:hint.

В качестве атрибута android:id вы можете использовать произвольные наименования.

SDK автоматически генерирует файл R.java при каждой компиляции приложения, поэтому вы не должны вручную изменять этот файл.

Создание строкового ресурса

По умолчанию строковые ресурсы хранятся в файле res/values/strings.xml. Добавим новый ресурс "edit_message" и укажем для него значение “Введите сообщение”.

1. В Android Studio откройте файл res/values/strings.xml
2. Добавьте строку с названием "edit_message" и значением “Введите сообщение”.
3. Добавьте строку с названием "button_send" и значением “Отправить”. Позже мы создадим кнопку, которая будет использовать данный ресурс.
4. Удалите строку, содержащую надпись "hello world".

Теперь файл strings.xml должен выглядеть следующим образом:

res/values/strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Мое первое приложение</string>
    <string name="edit_message">Введите сообщение</string>
    <string name="button_send">Отправить</string>
    <string name="action_settings">Настройки</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

Для надписей пользовательского интерфейса всегда используйте строковые ресурсы. Строковые ресурсы позволяют управлять надписями в одном месте, делают их проще для поиска и редактирования.

Более того, строковые ресурсы позволяют вам с легкостью перевести приложение на разные языки.

Добавление кнопки

1. В Android Studio откройте файл res/layout/activity_my.xml.
2. Создайте внутри <LinearLayout> элемент <Button> сразу после элемента <EditText>.
3. Задайте ширину как "wrap_content", чтобы она зависела от надписи внутри кнопки.
4. Добавьте атрибут android:text и укажите в качестве значения строковый ресурс "button_send", который мы создали в предыдущем параграфе.

Теперь <LinearLayout> должен выглядеть так:

res/layout/activity_my.xml

```
<LinearLayout xmlns:android="schemas.android.com/apk/res/android"
    xmlns:tools="schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
      <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Примечание: Мы не указываем для кнопки атрибут android:id, поскольку не собираемся обращаться к ней из кода.

Мы создали разметку, в которой оба виджета EditText и Button имеют размеры, соответсвующие их содержимому. Смотрите рисунок 2

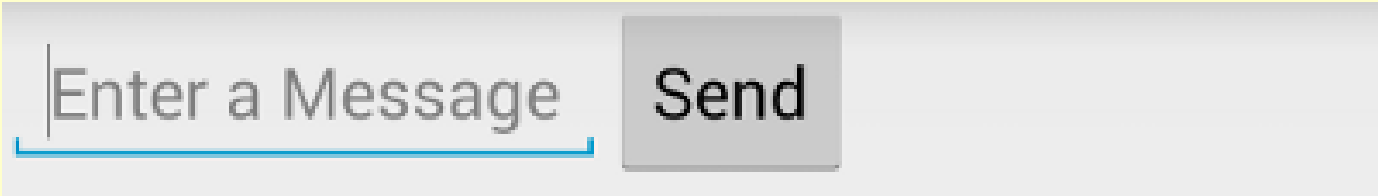


Рисунок 2.Кнопка и поле вводе с шириной равной "wrap_content".

Это удобно для кнопки, но плохо для текстового поля, потому что пользователь может вводить довольно длинную строку. Было бы неплохо растянуть текстовое поле на все неиспользуемое пространство экрана. При использовании LinearLayout это можно сделать, добавив атрибут веса android:layout_weight.

Вес – это число, показывающее количество пространства, которое может использовать каждый из элементов относительно друг друга. Это как в рецепте коктейля: “2 части водки, 1 часть сока” – это означает, что две трети коктейля состоит из водки. Рассмотрим пример. Если вы указываете для одного элемента значение 2, а для другого 1, сумма будет равна трем и первый элемент займет 2/3 свободного пространства, а второй заполнит остаток. Если вы добавите третий элемент и укажете для него вес, равный 1, то первый элемент (с весом 2) займет 1/2 пространства, а оставшиеся два элемента займут по 1/4.

Стандартный вес для всех элементов равен 0. Если вы укажете значение больше нуля только для одного элемента, то данный элемент заполнит все пространство, оставшееся после прорисовки других элементов.

Растягиваем поле ввода

Для того, чтобы растянуть элемент EditText на все свободное пространство проделаем следующее:

В файле activity_my.xml добавим элементу <EditText> атрибут layout_weight со значением 1, а атрибуту layout_width установим значение 0dp.

res/layout/activity_my.xml

```
<EditText
    android:layout_weight="1"
    android:layout_width="0dp"
    ... />
```

Мы указали ширину EditText равную 0dp. Установка нулевой ширины улучшает производительность разметки.
Использование "wrap_content" требует от системы лишнего вычисления ширины, результат которого в конечном счете не имеет смысла, поскольку указание веса все равно запустит операцию рассчета свободного пространства.

На рисунке 3 показан результат применения весового коэффициента к элементу EditText.

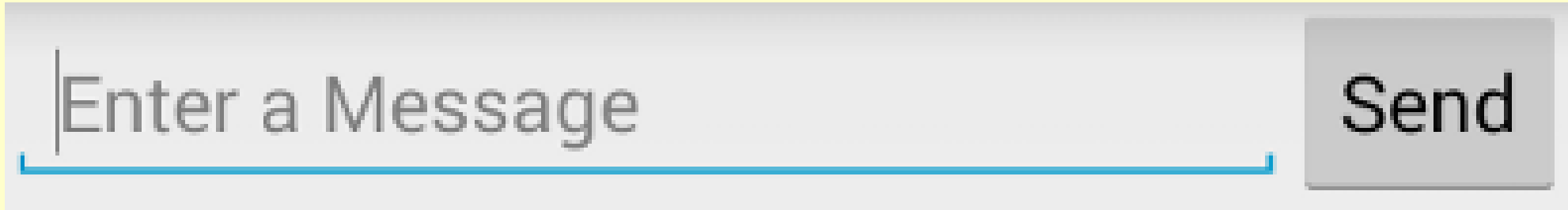


Рисунок 3.Вид поля ввода при установленом параметре weight.

Законченный вариант файла activity_my.xml выглядит таким образом:

res/layout/activity_my.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="schemas.android.com/apk/res/android"
    xmlns:tools="schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Запуск приложения

Созданная разметка используется классом Activity, который был сгенерирован SDK при создании проекта. Запустите приложение, чтобы посмотреть результат:

1. В Android Studio щелкните по кнопке Run на панели инструментов.
2. Если вы пользуетесь командной строкой, перейдите в корневую директорию вашего проекта и выполните команды:

```
1 ant debug
2 adb install bin/MyFirstApp-debug.apk
```

Запуск другого явления

После завершения предыдущего материала, у нас есть приложение, отображающее единственное явление, которое содержит текстовое поле ввода и кнопку. В данном занятии мы напишем код, который будет открывать новое явление при нажатии на кнопку.

Обработка нажатия на кнопку

Обработка нажатия на кнопку

1. В Android Studio откройте файл res/layout/activity_my.xml
2. Добавьте элементу <Button> атрибут android:onClick

res/layout/activity_my.xml

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Значение "sendMessage" атрибута android:onClick, это название метода в MyActivity, который будет вызван при нажатии на кнопку.

3. Откройте файл java/com.mycompany.myfirstapp/MyActivity.java

4. Добавьте метод sendMessage в класс MyActivity, как показано ниже

java/com.mycompany.myfirstapp/MyActivity.java

```
/** Данный метод вызывается при нажатии на кнопку */
public void sendMessage(View view) {
    // Обработка реакции нажатия
}
```

Для того, чтобы система сопоставила данный метод и имя метода, указанное в атрибуте android:onClick, метод должен удовлетворять следующим условиям:

- Быть публичным (public).
- Возвращать тип void.
- Иметь идинственный параметр типа View (в параметр передается объект View, который вызвал метод).

Далее мы напишем тело метода для передачи текста из поля ввода в другое явление.

Создание намерения (Intent)

1. В файле MyActivity.java создайте объект намерения типа Intent внутри метода sendMessage(). С помощью данного намерения мы будем запускать явление под названием DisplayMessageActivity.

java/com.mycompany.myfirstapp/MyActivity.java

```
public void sendMessage(View view) {
  Intent intent = new Intent(this, DisplayMessageActivity.class);
}
```

Примечание: Ссылка на DisplayMessageActivity вызовет ошибку в Android Studio, поскольку мы еще не создали класс с таким именем. Не беспокойтесь, скоро мы это сделаем.

Конструктор принимает два параметра:

- Первый параметр типа Context. Мы можем использовать this, поскольку класс Activity является наследником класса Context.
- Второй параметр типа Class указывает на компонент, к которому будет обращено данное намерение (В нашем случае явление, которое должно быть запущено).

Что такое намерение (Intents)

Намерение это объект, который выполняет связывание двух различных компонентов (например двух явлений). Намерения применяются для широкого спектра задач, но наиболее часто вы будете их использовать для запуска явлений. Подробную информацию смотрите в разделе Явления и фильтры.

2. Добавьте в начало файла импорт класса Intent:

java/com.mycompany.myfirstapp/MyActivity.java

import android.content.Intent;

Совет: В Android Studio нажмите Alt+Enter (option + return на Mac), чтобы импортировать недостающие классы.

3. Внутри sendMessage() создайте объект типа EditText, и воспользуйтесь методом findViewById() для получения ссылки на текстовое поле из нашей разметки.

java/com.mycompany.myfirstapp/MyActivity.java

```
public void sendMessage(View view) {
  Intent intent = new Intent(this, DisplayMessageActivity.class);
  EditText editText = (EditText) findViewById(R.id.edit_message);
}
```

4. Добавьте в начало файла импорт класса EditText.

5. Локальной переменной message присвойте значение из текстового поля и передайте это значение в намерение при помощи метода putExtra():

java/com.mycompany.myfirstapp/MyActivity.java

```
public void sendMessage(View view) {
  Intent intent = new Intent(this, DisplayMessageActivity.class);
  EditText editText = (EditText) findViewById(R.id.edit_message);
  String message = editText.getText().toString();
  intent.putExtra(EXTRA_MESSAGE, message);
}
```

Класс Intent позволяет передавать пары ключ-значение, называемые дополнительными данными.

Метод putExtra() принимает первым параметром ключ, а вторым параметром значение.

6. Объявите внутри класса MyActivity переменную EXTRA_MESSAGE следующим образом:

java/com.mycompany.myfirstapp/MyActivity.java

```
public class MyActivity extends ActionBarActivity {
    public final static String EXTRA_MESSAGE =
"com.mycompany.myfirstapp.MESSAGE";
    ...
}
```

Для того, чтобы другое явление смогло запросить дополнительные данные, необходимо сделать переменную с названием ключа публичной константой.

В целом использование названия пакета при объявлении ключей дополнительных данных, является хорошим стилем и мы рекомендуем его придерживаться.

Это гарантирует, что ключ будет уникальным при взаимодействии вашего приложения с любыми другими приложениями.

7. Внутри метода sendMessage() добавьте вызов startActivity() и передайте в качестве параметра созданный объект Intent.

Законченный метод sendMessage(), вызываемый при нажатии кнопки теперь выглядит так:

java/com.mycompany.myfirstapp/MyActivity.java

```
/** Вызывается при нажатии на кнопку */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

Система обрабатывает вызов и запускает экземпляр класса Activity, заданный в объекте типа Intent. Теперь чтобы все заработало, нам нужно создать класс DisplayMessageActivity.

Создание второго явления

Все наследники класса Activity должны реализовывать метод onCreate().

В данном методе явление получает и обрабатывает данные от намерений и выполняет первоначальную настройку всех компонентов явления.

Также в методе onCreate() должен вызываться метод setContentView() для выбора файла разметки данного явления.

Создание нового явления в Android Studio

Android Studio автоматически добавляет заготовку метода onCreate() при создании нового явления.

1. В Android Studio щелкните правой кнопкой мыши по пакету java/com.mycompany.myfirstapp и выберите New > Activity > Blank Activity.
2. В окне Choose options заполните поля:

- Activity Name (Название явления): DisplayMessageActivity
- Layout Name (Название разметки): activity_display_message
- Title (Заголовок): Моё сообщение
- Hierarchical Parent (Родительский элемент): com.mycompany.myfirstapp.MyActivity
- Package name (Название пакета): com.mycompany.myfirstapp

Нажмите Finish.

3. Откройте файл DisplayMessageActivity.java.

Как видите, обязательный метод onCreate() уже создан. Позже мы добавим в него нужный код. Также по умолчанию добавлен метод onOptionsItemSelected(), в котором описываются правила работы панели инструментов. Оставьте пока все как есть.

4. Удалите метод onCreateOptionsMenu(), в данном приложении он не понадобится.

Если вы работаете в Android Studio, то сейчас уже можете запустить обновленное приложение.

При нажатии на кнопку будет открываться второе явление, однако надпись в нем не будет меняться.

Дальше мы допишем наше новое явление, чтобы оно открывалось с текстом, введенным в поле ввода.

Создание явления без Android Studio

Для создания явления вы можете использовать любую другую среду разработки или командную строку.

1. Создайте новый файл DisplayMessageActivity.java в директории src/ вашего проекта.
2. Добавьте в файл следующий код:

```
public class DisplayMessageActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                .add(R.id.container, new PlaceholderFragment()).commit();
        }
    }
```

```
 @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      int id = item.getItemId();
      if (id == R.id.action_settings) {
         return true;
      }
      return super.onOptionsItemSelected(item);
   }
   public static class PlaceholderFragment extends Fragment {
      public PlaceholderFragment() { }
      @Override
      public View onCreateView(LayoutInflater inflater, ViewGroup container,
               Bundle savedInstanceState) {
          View rootView = inflater.inflate(R.layout.fragment_display_message,
                container, false);
          return rootView;
      }
   }
}
```

Примечание: Если вместо Android Studio вы используете какую-либо другую среду разработки, то в вашем проекте может не быть файла разметки activity_display_message, который передается первым параметром в setContentView().

Не волнуйтесь, позже мы исправим эту проблему.

3. Добавьте название нового явления в файл strings.xml:

```
<resources>
    ...
    <string name="title_activity_display_message">My Message</string>
</resources>
```

4. Добавьте в секцию <application> файла AndroidManifest.xml новый элемент <activity>. Это элемент для нашего класса DisplayMessageActivity.

```
<application ... >
    ...
    <activity
        android:name="com.mycompany.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="com.mycompany.myfirstapp.MyActivity" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.mycompany.myfirstapp.MyActivity" />
    </activity>
</application>
```

Атрибут android:parentActivityName указывает на родительский элемент явления.

Система использует эти данные для установки правил навигации внутри приложения, таких как Иерархическая навигация в Android 4.1 (уровень API 16) и выше. Вы можете создать такую же навигацию и для старых версий Android, используя библиотеку поддержки и добавив элемент <meta-data> в AndroidManifest.xml.

Примечание: Если вы следовали инструкции по установке дополнительных пакетов SDK, ваш Android SDK уже включает в себя последнюю версию библиотеки поддержки. При использовании Android Studio, библиотека поддержки автоматически добавляется к вашему проекту (вы можете увидеть jar файл в списке зависимостей). Если вы не используете Android Studio, вам нужно добавить файлы библиотеки вручную. Инструкцию вы можете посмотреть в разделе Добавление библиотеки поддержки.

Если вы используете стороннюю среду разработки, не переживайте, что ваш проект не компилируется. Сейчас мы добавим код для отображения введенного текста.

Получение намерения

Любое явление запускается с помощью намерения, независимо от используемых элементов навигации.

Вы можете получить экземпляр объекта намерения Intent, запустивший явление с помощью метода getIntent().

Экземпляр Intent будет содержать также все дополнительные данные.

1. Откройте файл java/com.mycompany.myfirstapp/DisplayMessageActivity.java.
2. В методе onCreate() удалите следующую строку:

```
setContentView(R.layout.activity_display_message);
```

3. Создайте переменную типа Intent и присвойте ей значение getIntent().
4. В верхней части файла импортируйте класс Intent.

В Android Studio нажмите сочетание клавиш Alt+Enter (option + return на Mac) для импорта недостающих классов.

5. Получить дополнительные данные строкового типа, переданные из MyActivity, можно с помощью метода getStringExtra().

```
String message = intent.getStringExtra(MyActivity.EXTRA_MESSAGE);
```

Отображение сообщения

1. Создайте объект типа TextView в методе onCreate()

```
TextView textView = new TextView(this);
```

2. Установите размер шрифта и сообщение с помощью метода setText():

```
textView.setTextSize(40);

textView.setText(message);
```

3. Мы можем сделать объект типа TextView корневым элементом разметки явления. Для этого передадим объект в качестве параметра метода setContentView().

```
setContentView(textView);
```

4. Импортируйте класс TextView.

В Android Studio нажмите сочетание клавиш Alt+Enter (option + return на Mac) для импорта недостающих классов.

Теперь метод onCreate() класса DisplayMessageActivity должен выглядеть следующим образом:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Получить сообщение из Intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MyActivity.EXTRA_MESSAGE);
    // Создание объекта TextView
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);
    // Делаем textView корневым элементов разметки activity
    setContentView(textView);
}
```

Теперь вы можете запустить приложение. После запуска введите любой текст в поле ввода и нажмите кнопку Отправить.

Сообщение откроется во втором явлении.

Поздравляем! Вы создали свое первое приложение под Android!

Чтобы узнать больше о разработке для Android, начните изучение следующего материала Добавление панели инструментов.

Задание

1. Создать простой проект в Android Studio
2. Произвести запуск приложения
3. Создать простой интерфейс пользователя. Создать линейную разметку (Linear Layout). Добавить текстовые поля. Создать строковый ресурс. Добавить кнопки. Растянуть поле ввода. Запустить другое явление. Реализовать обработку нажатия на кнопку. Создать намерения (Intent)
4. Создать второе явление. Создать новое явление в Android Studio. Создать явления без Android Studio. Обеспечить получение намерения. Обеспечить отображение сообщения