Навчальна дисципліна

Розробка мобільних застосувань

Лектор - к.т.н., доцент

Баклан Ігор Всеволодович

Site: baklaniv.at.ua

E-mail: iaa@ukr.net

2016-2017

Лекція №1. Вступ до програмування в Android

Система Android покорила мир. Все хотят иметь планшет или смартфон, а устройства на базе Android пользуются невероятной популярностью. В этой книге мы научим вас разрабатывать собственные приложения, а также покажем, как построить простое приложение и запустить его на виртуальном устройстве Android. Попутно будут рассмотрены основные компоненты приложений Android — такие как активности и макеты. Все, что от вас потребуется — некоторые базовые знания Java...

Добро пожаловать в мир Android

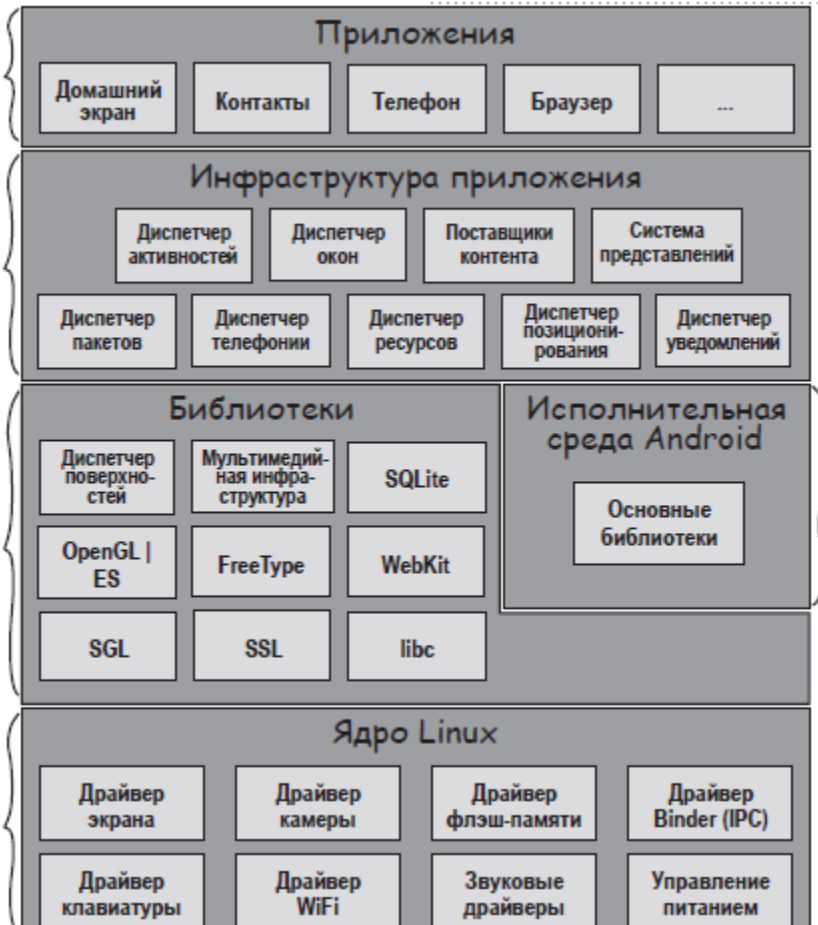
Android — самая популярная мобильная платформа в мире. Согласно последним опросам, в мире свыше *миллиарда* активных Android-устройств, и их количество продолжает стремительно расти. Android — полнофункциональная платформа с открытым кодом на базе Linux, разрабатываемая компанией Google. Это мощная платформа разработки, включающая все необходимое для построения современных приложений из кода Java и XML. Более того, построенные приложения могут устанавливаться на множестве разных устройств — телефонах, планшетах и не только. Что же собой представляет типичное Android-приложение?

Android включает несколько базовых приложений — таких, как Контакты, Календарь, Карты и браузер.

При построении приложений вам доступны те же API, которые используются базовыми приложениями. При помощи этих API вы управляете внешним видом и поведением своих приложений.

Под инфраструктурой приложений располагается уровень библиотек C и C++. Для работы с ними используются API.

В самом основании системы лежит ядро Linux. В Android оно обеспечивает работу драйверов, а также таких базовых сервисов, как безопасность и управление памятью.



Исполнительная среда Android включает набор базовых библиотек, реализующих большую часть языка программирования Java. Каждое Android-приложение выполняется в отдельном процессе.

Вот что мы сейчас сделаем

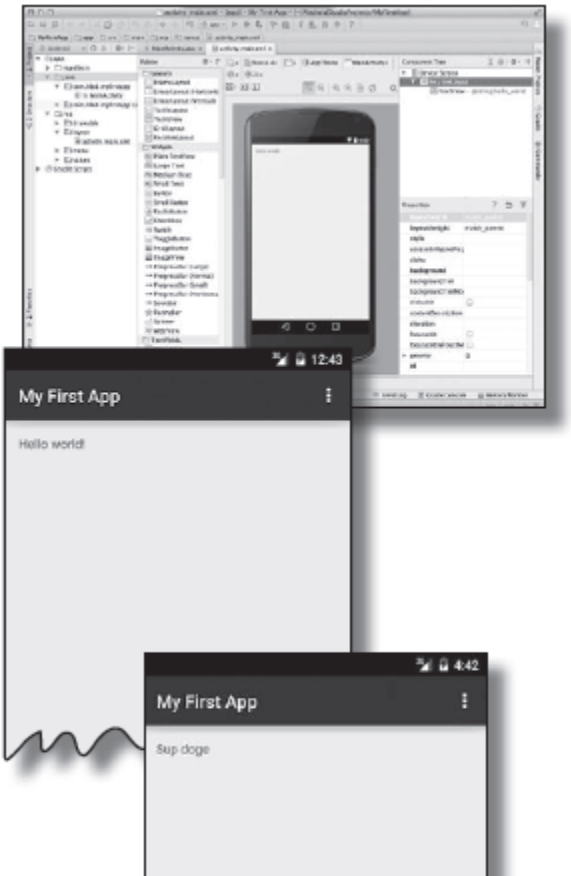
Давайте сходу возьмемся за дело и построим простейшее Android-приложение. Для этого необходимо выполнить лишь несколько действий:

1. **Подготовка среды разработки.**
   Необходимо установить среду Android Studio, включающую все необходимое для разработки Android-приложений.

2. **Построение простейшего приложения.**
   Мы создадим в Android Studio простое приложение, которое будет выводить текст на экран.

3. **Запуск приложения в эмуляторе Android.**
   Мы воспользуемся встроенным эмулятором, чтобы увидеть приложение в действии.

4. **Изменение приложения.**
   Наконец, мы внесем несколько изменений в приложение, созданное на шаге 2, и снова запустим его.



Вы находитесь здесь.

- [ ] Подготовка среды
- [ ] Построение приложения
- [ ] Запуск приложения
- [ ] Изменение приложения

Среда разработки

Java — самый популярный язык, используемый для разработки Android-приложений. Устройства на базе Android не запускают файлы *.class* и *.jar*. Вместо этого для повышения скорости и эффективности использования аккумуляторов Android-устройства используют собственные оптимизированные форматы компилированного кода. Это означает, что вы не сможете воспользоваться обычной средой разработки на языке Java — вам также понадобятся специальные инструменты для преобразования откомпилированного кода в формат Android, установки его на Android-устройствах и отладки приложения, когда оно заработает. Все необходимое содержится в **Android SDK**. Посмотрим, что в него входит.

Android SDK

Пакет Android Software Development Kit (SDK) содержит библиотеки и инструменты, необходимые для разработки Android-приложений:

SDK Platform
Отдельная платформа для каждой версии Android.

SDK Tools
Инструменты отладки и тестирования, а также другие полезные служебные программы. Также включает набор платформенно-зависимых инструментов.

Примеры приложений
Если вы захотите просмотреть реальные примеры кода, чтобы лучше понять, как пользоваться API, примеры приложений вам в этом помогут.

Лишь некоторые компоненты, входящие в пакет.

Документация
Предоставляет автономный доступ к новейшей документации API.

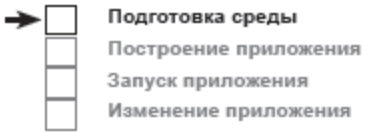
Вспомогательный инструментарий Android
Дополнительные API, отсутствующие в стандартной платформе.

Google Play Billing
Интеграция сервиса биллинга в приложения.

Android Studio — специализированная версия IntelliJ IDEA

IntelliJ IDEA — одна из самых популярных интегрированных сред разработки (IDE) для программирования на Java. Android Studio — версия IDEA, которая включает версию Android SDK и дополнительные инструменты графических интерфейсов, упрощающие разработку приложений.

Кроме редактора и доступа к инструментам и библиотекам из Android SDK, Android Studio предоставляет шаблоны, упрощающие создание новых приложений и классов, а также средства для выполнения таких операций, как упаковка приложений и их запуск.



Установите Java

Android Studio — среда разработки на языке Java, поэтому на вашем компьютере должна быть установлена правильная версия Java. Сначала проверьте системные требования Android Studio и определите, какие версии Java Development Kit (JDK) и Java Runtime Edition (JRE) вам понадобятся. Системные требования можно просмотреть здесь:

http://developer.android.com/sdk/index.html#Requirements

Oracle и Google иногда меняют свои URL-адреса. Если эти не работают, выполните поиск.

Когда вы будете знать, какие версии JDK и JRE вам понадобятся, загрузите и установите их отсюда:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Затем установите Android Studio

Если этот URL-адрес изменится, поищите Android Studio на сайте developer.android.com.

Когда Java успешно заработает, загрузите Android Studio отсюда:

https://developer.android.com/sdk/installing/index.html?pkg=studio

На этой странице также размещены инструкции по установке. Выполните их, чтобы установить Android Studio на вашем компьютере. Когда установка будет завершена, откройте Android Studio и выполните инструкции по добавлению новейших инструментов SDK и библиотек поддержки. Когда все будет сделано, на экране появляется заставка Android Studio. Все готово для построения вашего первого Android-приложения.

Мы не приводим инструкции по установке, потому что они довольно быстро устаревают. Следуйте инструкциям из электронной документации, и все будет нормально.

Построение простого приложения

Итак, среда разработки подготовлена, и можно переходить к созданию вашего первого Android-приложения. Вот как оно будет выглядеть:

Этот шаг завершен, мы его вычеркиваем.

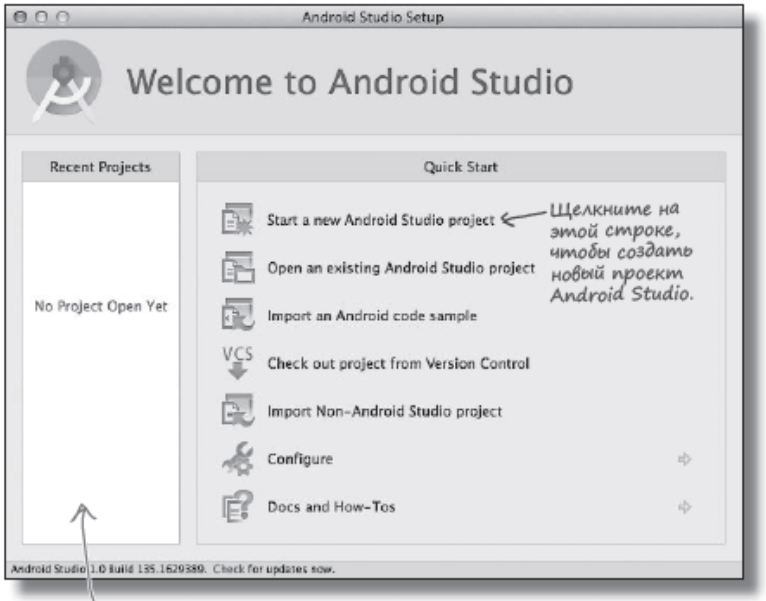
- [x] Подготовка среды
- [ ] **Построение приложения**
- [ ] Запуск приложения
- [ ] Изменение приложения

Давайте построим простое приложение

Каждый раз, когда вы создаете новое приложение, для него необходимо создать новый проект. Убедитесь в том, что среда Android Studio открыта, и повторяйте за нами.

1. Создание нового проекта

На заставке Android Studio перечислены некоторые возможные операции. Мы хотим создать новый проект; щелкните на строке "Start a new Android Studio project".



Построение простого приложения (продолжение)

2. Настройка проекта

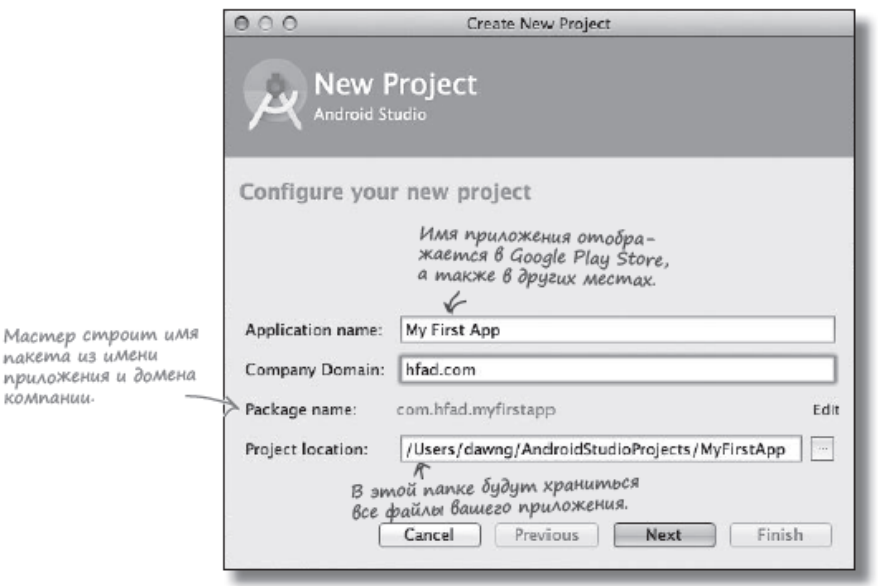
Теперь необходимо настроить конфигурацию приложения: указать, как оно будет называться, какой домен компании будет использоваться и где должны храниться его файлы. Android Studio использует домен компании и имя приложения для формирования имени пакета, которое будет использоваться вашим приложением. Например, если присвоить приложению имя "My First App" и указать домен компании "hfad.com", то Android Studio сформирует имя пакета `com.hfad.myfirstapp`. Имя пакета играет очень важную роль в Android, потому что оно используется Android-устройствами для *однозначной идентификации приложения*.

Введите имя приложения "My First App", домен компании "hfad.com", и подтвердите местоположение по умолчанию. Щелкните на кнопке Next.

Будьте осторожны!

Имя пакета должно оставаться неизменным на протяжении всего срока жизни приложения.

Это уникальный идентификатор вашего приложения, который используется для управления версиями программы.



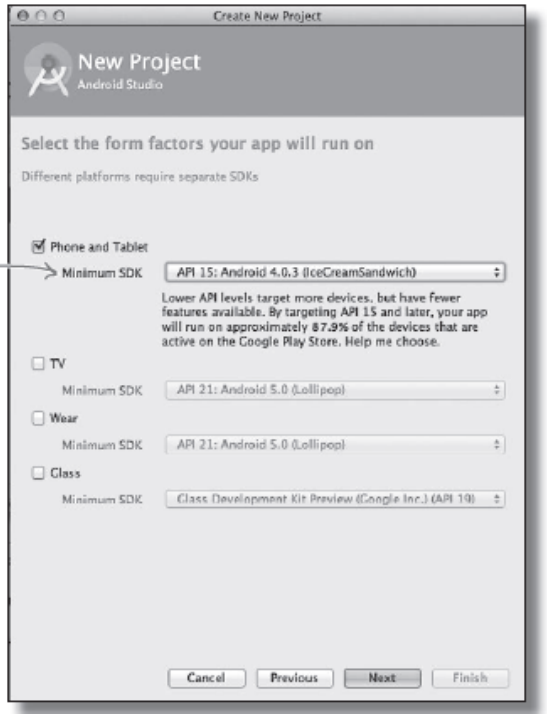
Построение простого приложения (продолжение)

3. Выбор уровня API

Теперь необходимо указать, какие уровни API системы Android будут использоваться вашим приложением. Уровни API увеличиваются с выходом каждой очередной версии Android. Если только вы не хотите, чтобы приложение работало только на самых новых устройствах, стоит выбрать один из более старых уровней API. Здесь мы выбираем API уровня 15; это означает, что приложение сможет работать на большинстве устройств. Кроме того, наша версия приложения создается только для телефонов и планшетов, поэтому флажки остальных вариантов так и остаются снятыми. Когда это будет сделано, щелкните на кнопке Next.

Дополнительная информация о разных уровнях API приведена на следующей странице.

Минимальный уровень SDK — наименьшая версия, которая будет поддерживаться вашим приложением. Приложение будет работать на устройствах с API этого уровня и выше. На устройствах с API более низкого уровня оно работать не будет.



Вероятно, вам не раз доводилось слышать, как применительно к Android упоминаются разные «вкусные» названия: Ice Cream Sandwich (сэндвич с мороженым), Jelly Bean (мармеладная конфета), KitKat и Lollipop (леденец). Что это за кондитерская?

Каждой версии Android присваивается номер и кодовое имя. Номер версии определяет конкретную версию Android (например, 5.0), тогда как кодовое имя представляет собой чуть более общее «дружественное» имя, которое может объединять сразу несколько версий Android. Под «уровнем API» понимается версия API, используемых приложением. Например, Android версии 5.0 соответствует уровень API 21.

Версия	Кодовое имя	Уровень API
1.0		1
1.1		2
1.5	Cupcake	3
1.6	Donut	4
2.0	Eclair	5
2.01	Eclair	6
2.1	Eclair	7
2.2.x	Froyo	8
2.3 — 2.3.2	Gingerbread	9
2.3.2 — 2.3.7	Gingerbread	10
3.0	Honeycomb	11
3.1	Honeycomb	12
3.2	Honeycomb	13
4.0 — 4.0.2	Ice Cream Sandwich	14
4.0.3-4.0.4	Ice Cream Sandwich	15
4.1	Jelly Bean	16
4.2	Jelly Bean	17
4.3	Jelly Bean	18
4.4	KitKat	19
4.4	KitKat (with wearable extensions)	20
5.0	Lollipop	21

Сейчас эти версии уже не встречаются.

На большинстве устройств используется один из этих уровней API.

При разработке Android-приложений необходимо учитывать, с какими версиями Android должно быть совместимо ваше приложение. Если вы укажете, что приложение совместимо только с самой последней версией SDK, может оказаться, что оно не запускается на очень многих устройствах. Информацию о процентном распределении версий по устройствам можно найти здесь: *https://developer.android.com/about/dashboards/index.html*.

Активности и макеты: с высоты птичьего полета

Далее вам будет предложено добавить активность в ваш проект. Каждое Android-приложение состоит из экранов, а каждый экран состоит из активности и макета.

Активность — одна четко определенная операция, которую может выполнить пользователь. Например, в приложении могут присутствовать активности для составления сообщения электронной почты, поиска контакта или создания снимка. Активности обычно ассоциируются с одним экраном и программируются на Java.

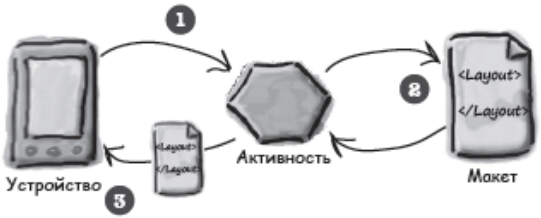
Макет описывает **внешний вид экрана.** Макеты создаются в виде файлов в разметке XML и сообщают Android, где располагаются те или иные элементы экрана.

Рассмотрим подробнее, как взаимодействуют активности и макеты.

Макеты определяют способ представления пользовательского интерфейса.

Активности определяют действия.

1. Устройство запускает приложение и создает объект активности.
2. Объект активности задает макет.
3. Активность приказывает Android вывести макет на экран.
4. Пользователь взаимодействует с макетом, отображаемым на устройстве.
5. Активность реагирует на эти взаимодействия, выполняя код приложения.
6. Активность обновляет содержимое экрана...
7. ...и пользователь видит это на устройстве.

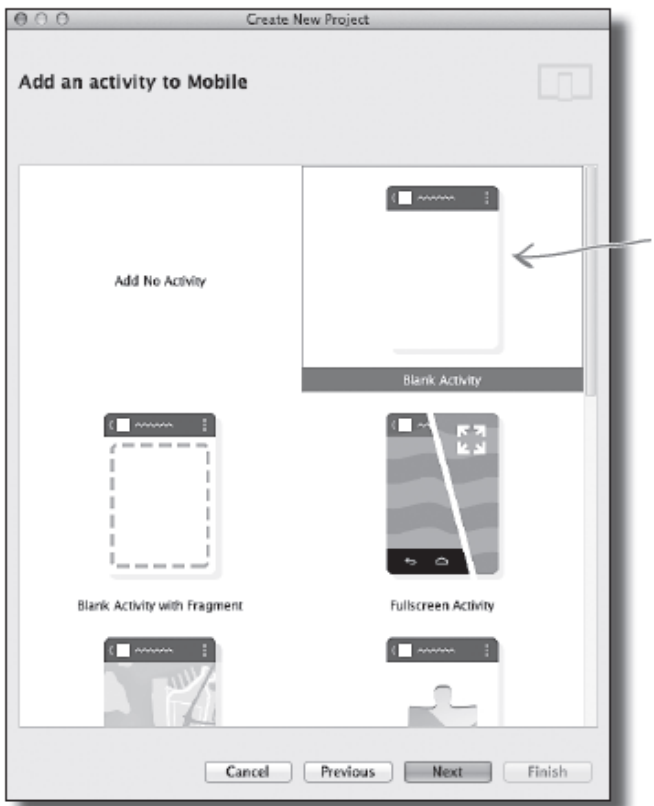


Теперь, когда вы чуть больше знаете о том, что собой представляют активности и макеты, мы пройдем два последних шага мастера и прикажем ему создать простейшую активность и макет.

Построение простого приложения (продолжение)

4. Создание активности

На следующем экране представлен набор шаблонов, которые могут использоваться для создания активности и макета. Вы должны выбрать один из них. Так как в нашем приложении будут использоваться простейшая активность и макет, выберите вариант Blank Activity и щелкните на кнопке Next.



Есть и другие типы активностей, которые можно выбрать в этом окне, но на этот раз следует выбрать вариант Blank Activity.

Построение простого приложения (продолжение)

5. Настройка активности

Теперь среда разработки предложит выбрать имена для активности и макета экрана. Вам также придется указать текст заголовка экрана и имя ресурса меню. Введите имя активности "MainActivity" и имя макета "activity_main". Активность представляет собой класс Java, а макет — файл с разметкой XML, поэтому для введенных нами имен будет создан файл класса Java с именем *MainActivity.java* и файл XML с именем *activity_main.xml*.
Когда вы щелкнете на кнопке Finish, Android Studio построит приложение.

Присвойте активности имя "MainActivity", а макету — имя "activity_main". Для остальных параметров подтвердите значения по умолчанию.

Только что вы создали свое первое Android-приложение

Подготовка среды
Построение приложения
Запуск приложения
Изменение приложения

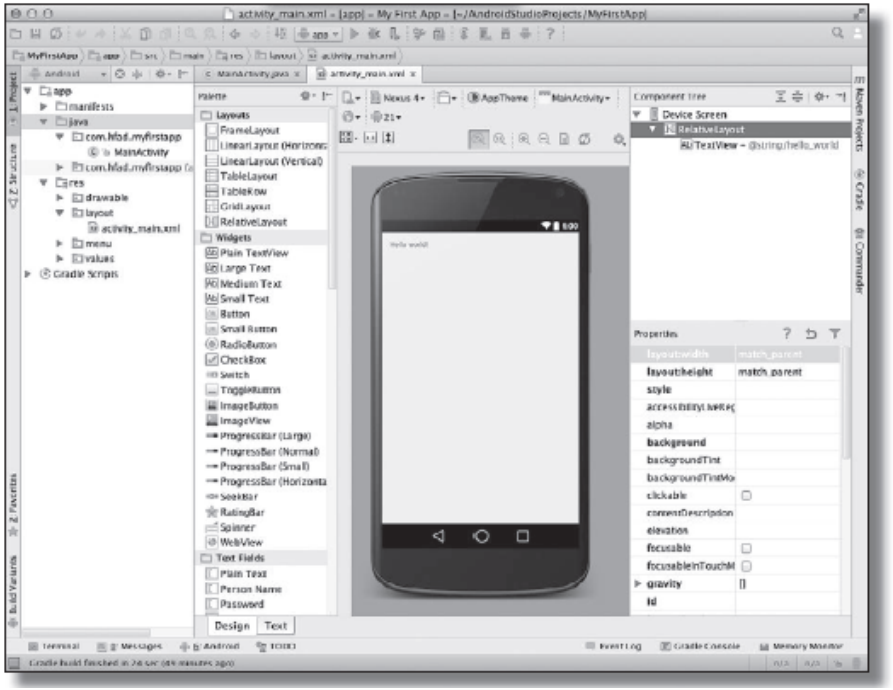
Итак, что же произошло?

- **Мастер Android Studio создал для вашего приложения проект, параметры которого были настроены в соответствии с вашими указаниями.**
  Вы определили, с какими версиями Android должно быть совместимо приложение, а мастер создал все файлы и папки, необходимые для простейшего работоспособного приложения.

- **Мастер создал базовую активность и базовый макет с шаблонным кодом.**
  Шаблонный код включает разметку XML для макета и код Java для активности; он выводит текст "Hello world!" в макете. Вы можете изменить этот код.

Когда вы завершите создание проекта, пройдя все шаги работы с мастером, Android Studio автоматически отобразит проект в среде разработки.
Вот как выглядит наш проект на текущий момент (не беспокойтесь, если сейчас все выглядит слишком сложно — на нескольких ближайших страницах мы все объясним):

Так выглядит проект в Android Studio.



Android Studio создает всю структуру папок за вас

Android-приложение в действительности представляет собой набор файлов, размещенных в четко определенной структуре папок; Android Studio создает все эти папки за вас при создании нового приложения. Если вас интересует, как выглядит эта структура папок, проще всего просмотреть ее на панели у левого края окна Android Studio. На ней отображаются все проекты, открытые в настоящее время. Чтобы свернуть или развернуть содержимое папки, щелкните на стрелке слева от значка папки.

В структуре папок присутствуют файлы разных типов

Просмотрев структуру папок, вы увидите, что мастер создал за вас папки и файлы разных типов:

Исходные файлы Java и XML
Файлы активности и макета, которые были созданы за вас мастером.

Файлы Java, сгенерированные Android
Дополнительные файлы Java, которые Android Studio тоже генерирует автоматически. Вносить в них изменения вам не придется.

Файлы ресурсов
К этой категории относятся файлы изображений на значках по умолчанию, стили, которые могут использоваться вашим приложением, и все общие строковые данные, к которым может обращаться приложение.

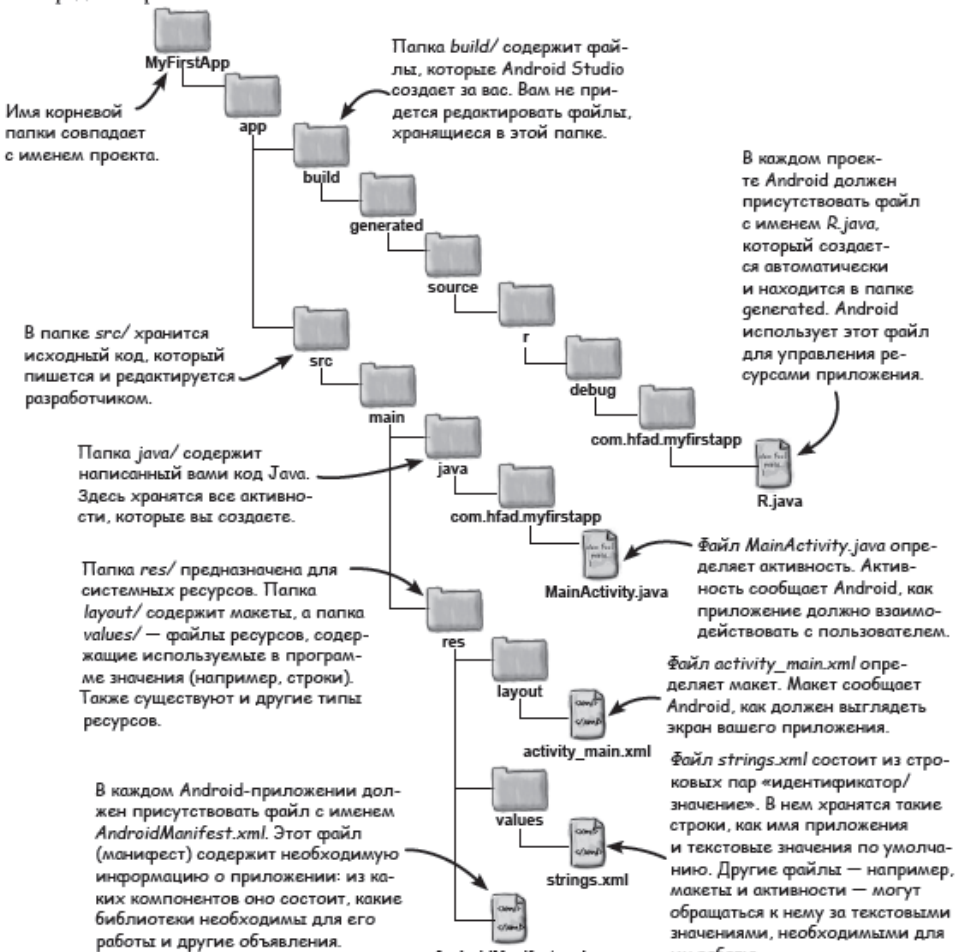
Библиотеки Android
В окне мастера вы указали минимальную версию SDK, с которой должно быть совместимо ваше приложение. Android Studio включает в приложение библиотеки Android, актуальные для этой версии.

Файлы конфигурации
Файлы конфигурации сообщают Android, что содержит приложение и как его следует выполнять.

Давайте внимательнее присмотримся к некоторым ключевым файлам и папкам в мире приложений Android.

Полезные файлы в проекте

Проекты Android Studio используют систему сборки gradle для компиляции и развертывания приложений. Проекты gradle имеют стандартную структуру. Некоторые ключевые файлы и папки, с которыми вам предстоит работать:



Редактирование кода в Android Studio

Для просмотра и изменения файлов используются различные редакторы Android Studio. Сделайте двойной щелчок на файле, с которым вы хотите работать; его содержимое появляется в середине окна Android Studio.

Редактор кода

Большинство файлов отображается в редакторе кода. По сути это обычный текстовый редактор, но с поддержкой таких дополнительных возможностей, как цветовое выделение синтаксиса и проверка кода.

Сделайте двойной щелчок на файле...

...содержимое файла открывается на панели редактора.

Визуальный редактор

При редактировании макета появляется дополнительная возможность: вместо редактирования разметки XML можно использовать визуальный редактор. Визуальный редактор позволяет перетащить компоненты графического интерфейса на макет и расположить их так, как вы считаете нужным. Редактор кода и визуальный редактор обеспечивают разные представления одного файла, и вы можете переключаться между ними по своему усмотрению.

При помощи этих вкладок вы выбираете используемый редактор.

Это разные представления одного файла. В одном представлении отображается код, а в другом — визуальный результат.

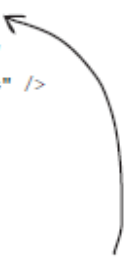
Ниже приведен фрагмент кода из файла макета, сгенерированного Android Studio. Да, мы знаем, что вы еще ни разу не видели код макета, и все же попробуйте соединить каждое из описаний в нижней части страницы с правильной строкой кода. Мы уже провели одну линию, чтобы вам было проще взяться за дело.

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
android:paddingBottom="16dp"
tools:context=".MainActivity">

<TextView
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</RelativeLayout>
```



Добавить отступы у краев экрана.

Добавить графический компонент `TextView` (надпись) для вывода текста.

Включить перенос текста по горизонтали и вертикали.

Вывести значение ресурсной строки с именем `hello_world`.

Назначить высоту и ширину макета по размерам экрана устройства.

Ниже приведен фрагмент кода из файла макета, сгенерированного Android Studio. Да, мы знаем, что вы еще ни разу не видели код макета, и все же попробуйте соединить каждое из описаний в нижней части страницы с правильной строкой кода. Мы уже провели одну линию, чтобы вам было проще взяться за дело.

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
android:paddingBottom="16dp"
tools:context=".MainActivity">

<TextView
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</RelativeLayout>
```

А теперь посмотрим, удастся ли вам сделать то же с кодом активности. **Это условный код, а не тот код, который Android Studio генерирует за вас.** Соедините каждое описание с правильной строкой кода.

MainActivity.java

```
package com.hfad.myfirstapp;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Имя пакета.

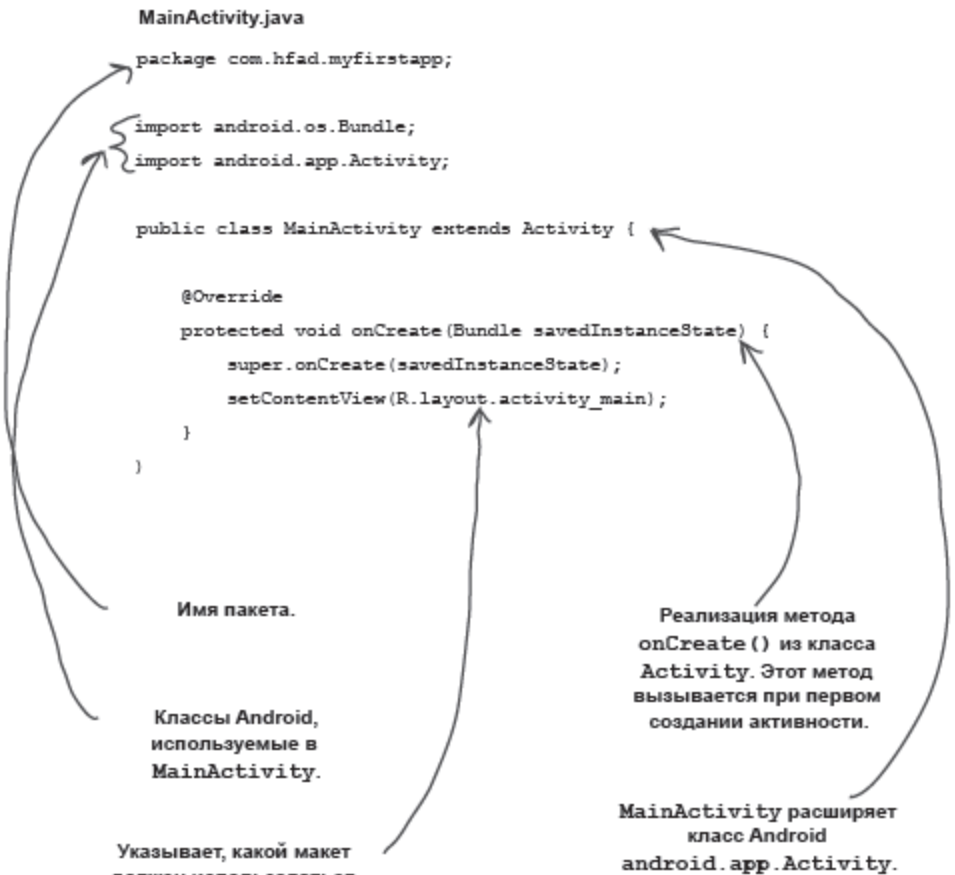
Классы Android, используемые в `MainActivity`.

Указывает, какой макет должен использоваться.

Реализация метода `onCreate()` из класса `Activity`. Этот метод вызывается при первом создании активности.

`MainActivity` расширяет класс Android `android.app.Activity`.

А теперь посмотрим, удастся ли вам сделать то же с кодом активности. **Это условный код, а не тот код, который Android Studio генерирует за вас.** Соедините каждое описание с правильной строкой кода.



Запуск приложения в эмуляторе Android

Итак, вы увидели, как Android-приложение выглядит в Android Studio, и в общих чертах представили, как работает система в целом. Но *на самом деле* вам хочется увидеть, как работает приложение, верно?

В том, что касается запуска приложений, есть пара вариантов. Вариант первый — запустить приложение на физическом устройстве. Но что, если у вас нет такого устройства под рукой? Или вы хотите узнать, как оно будет выглядеть на устройстве другого типа, которого у вас вообще нет?

Альтернативное решение — воспользоваться **эмулятором Android,** встроенным в Android SDK. Эмулятор позволяет создать одно или несколько **виртуальных устройств Android** (Android Virtual Device, AVD) и запустить приложение в эмуляторе *так, словно оно выполняется на физическом устройстве.*

Эмулятор Android позволяет запустить приложение на виртуальном устройстве Android (AVD). AVD ведет себя практически так же, как и физическое Android-устройство. Вы можете создать сразу несколько AVD для разных типов устройств.

Как же выглядит эмулятор?

Перед вами AVD в эмуляторе Android. Оно выглядит как телефон, работающий на вашем компьютере.

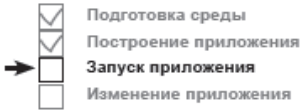
Эмулятор представляет собой приложение, точно воссоздающее аппаратное окружение устройства Android: от центрального процессора и памяти до звуковых микросхем и экрана. Эмулятор построен на базе существующего эмулятора QEMU, похожего на другие виртуальные машины, с которыми вам, возможно, доводилось работать — такие, как VirtualBox или VMWare.

Внешний вид и поведение AVD зависят от заданных вами параметров. На иллюстрации AVD эмулирует Nexus 4, поэтому эмулятор выглядит и ведет себя так, словно на вашем компьютере имеется внутреннее устройство Nexus 4.

Давайте создадим AVD, чтобы вы могли увидеть свое приложение, выполняемое в эмуляторе.

После создания AVD вы увидите свое приложение, выполняемое на нем. Android Studio запускает эмулятор автоматически.

Как и физический телефон, AVD необходимо разблокировать перед началом использования. Просто щелкните на значке с замком и перетащите вверх.



Создание виртуального устройства Android

Создание AVD в Android Studio состоит из нескольких шагов. Мы создадим AVD для Nexus 4 с уровнем API 21, чтобы вы могли видеть, как ваше приложение выглядит и ведет себя на устройствах этого типа. Последовательность действий остается более или менее постоянной для любого типа устройств.

Откройте AVD Manager

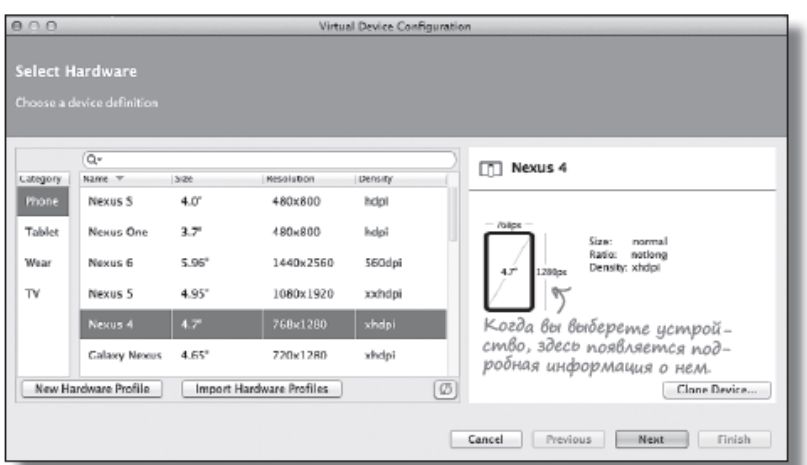
В диспетчере AVD Manager вы сможете создавать новые AVD, а также просматривать и редактировать уже созданные виртуальные устройства. Чтобы запустить его, выберите в меню Tools пункт Android и выберите AVD Manager.

Если вы еще не создали ни одного виртуального устройства, открывается окно с предложением создать его. Щелкните на кнопке "Create a virtual device".

Щелкните на кнопке "Create a virtual device", чтобы создать AVD.

Выберите тип устройства

На следующем экране вам будет предложено выбрать определение устройства — то есть тип устройства, который будет эмулировать AVD. Давайте посмотрим, как будет выглядеть наше приложение на телефоне Nexus 4. Выберите в меню Category пункт Phone, затем выберите в списке Nexus 4. Щелкните на кнопке Next.



Создание виртуального устройства Android (продолжение)

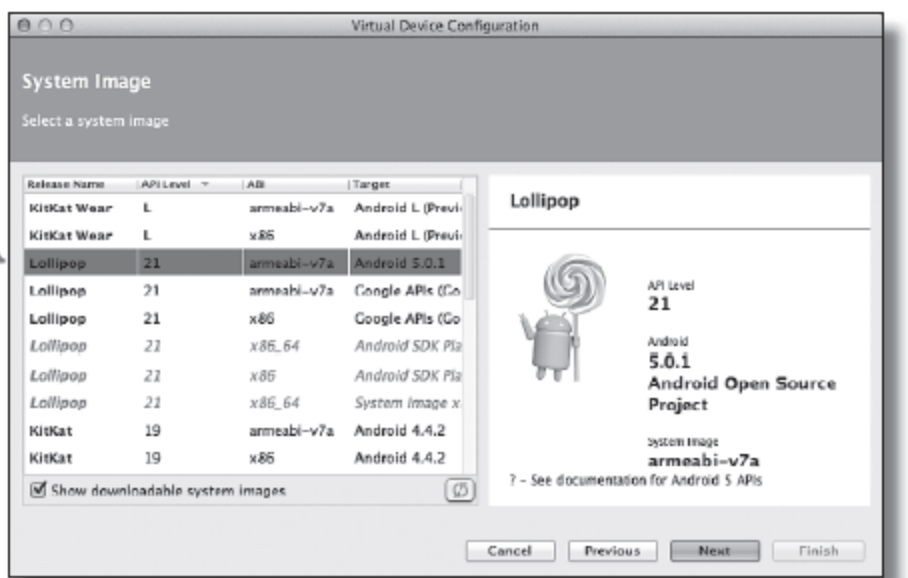
Подготовка среды
Построение приложения
Запуск приложения
Изменение приложения

Выберите образ системы

Далее следует выбрать образ системы, то есть установленную версию операционной системы Android. Вы можете выбрать версию Android, которая должна поддерживаться AVD, и тип процессора (ARM или x86).

Вы должны выбрать образ системы для уровня API, совместимого с создаваемым приложением. Например, если вы хотите, чтобы приложение работало на минимальном уровне API 15, выберите образ системы с уровнем API *не менее* 15. Мы будем использовать образ системы для уровня API 21. Выберите строку Lollipop/21/armeabi-v7a с целевой системой Android 5.0.1. Щелкните на кнопке Next.

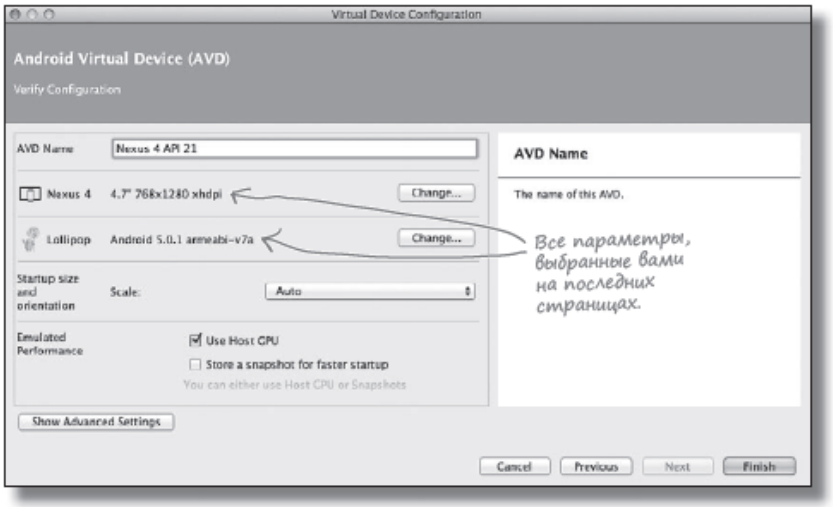
Если этот образ системы не установлен на вашем компьютере, вам будет предоставлена возможность загрузить его.



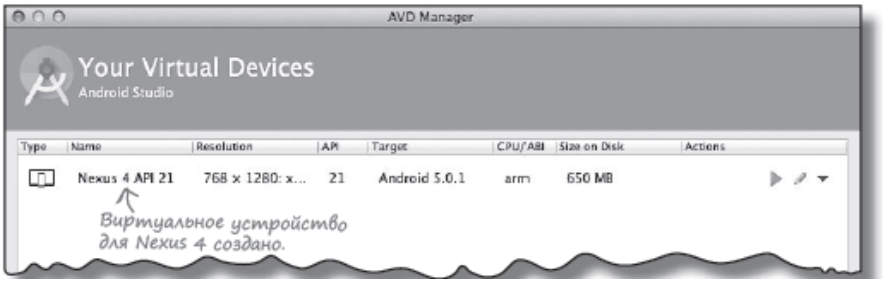
Создание виртуального устройства Android (продолжение)

Проверка конфигурации AVD

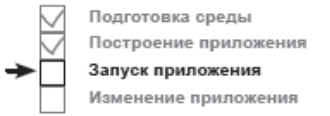
На следующем экране вам будет предложено подтвердить конфигурацию AVD. На нем приведена сводка параметров, выбранных вами на нескольких последних экранах, а также предоставляется возможность изменить их. Подтвердите значения и щелкните на кнопке Finish.



AVD Manager создает AVD и, когда виртуальное устройство будет создано, отображает его в списке устройств. Теперь AVD Manager можно закрыть.



Запуск приложения в эмуляторе



Теперь, когда вы создали виртуальное устройство, используйте его для запуска приложения. Для этого выберите команду "Run 'app'" из меню Run. Когда вам будет предложено выбрать устройство, убедитесь в том, что установлен переключатель "Launch emulator" с только что созданным вами виртуальным устройством Nexus 4 AVD. Щелкните на кнопке OK.

Пока вы терпеливо ожидаете появления AVD, посмотрим, что происходит при выполнении команды Run.

Виртуальное устройство Android, которое мы только что создали.

Компиляция, упаковка, развертывание и запуск

Команда Run не просто запускает приложение. Она также выполняет все подготовительные операции, необходимые для запуска приложения:

Файл APK — файл пакета приложения Android. По сути это архив JAR или ZIP с приложением Android.

1. **Файлы с исходным кодом Java компилируются в байт-код.**

2. **Создается пакет Android-приложения, или файл APK.**
   Файл APK включает откомпилированные файлы Java, а также все библитеки и ресурсы, необходимые для работы приложения.

3. **Если эмулятор не выполняется в настоящий момент, он запускается с AVD.**

4. **Когда эмулятор будет запущен, а AVD активизируется, файл APK передается на AVD и устанавливается.**

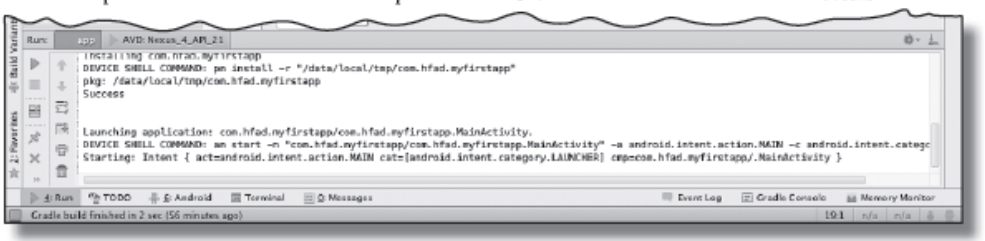
5. **AVD запускает главную активность, связанную с приложением.**
   Ваше приложение отображается на экране AVD. Теперь ничто не мешает тому, чтобы проверить его в работе.

Информация о ходе запуска отображается на консоли

Запуск эмулятора с AVD может занимать немало времени — обычно *несколько минут*. К счастью, за ходом операции можно проследить на консоли Android Studio. На консоли выводится подробный отчет о том, что делает система сборки gradle, а если в процессе запуска возникнут какие-либо ошибки — они будут выделены в тексте.

Консоль располагается в нижней части экрана Android Studio:

Пока эмулятор запускается, вы можете заняться чем-нибудь полезным: скажем, вышиванием или приготовлением обеда.



Android Studio запускает эмулятор с AVD Nexus4, только что созданным нами виртуальным устройством Android.

Вот что выводится в окне консоли при запуске приложения:

```
Waiting for device.
/Applications/adt-bundle-mac/sdk/tools/emulator -avd Nexus_4_API_21 -netspeed full -netdelay none
Device connected: emulator-5554
Device Nexus_4_API_21 [emulator-5554] is online, waiting for processes to start up..
Device is ready: Nexus_4_API_21 [emulator-5554]
Target device: Nexus_4_API_21 [emulator-5554]
Uploading file
        local path: /Users/dawng/AndroidStudioProjects/MyFirstApp/app/build/outputs/apk/app-debug.apk
        remote path: /data/local/tmp/com.hfad.myfirstapp
Installing com.hfad.myfirstapp
DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/com.hfad.myfirstapp"
pkg: /data/local/tmp/com.hfad.myfirstapp
Success
Launching application: com.hfad.myfirstapp/com.hfad.myfirstapp.MainActivity.
DEVICE SHELL COMMAND: am start -n "com.hfad.myfirstapp/com.hfad.myfirstapp.MainActivity" -a
android.intent.action.MAIN -c android.intent.category.LAUNCHER
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=com.hfad.myfirstapp/.MainActivity }
```

Виртуальное устройство Android запущено и готово к работе.

Загрузка и установка файла APK.

Наконец, наше приложение начинает работу с запуска его главной активности — той самой, которую мастер сгенерировал за вас.

Тест-драйв

- [x] Подготовка среды
- [x] Построение приложения
- [x] Запуск приложения
- [ ] Изменение приложения

Итак, посмотрим, что же на самом деле происходит на экране при запуске приложения.
Сначала эмулятор запускается в отдельном окне. Эмулятор довольно долго загружает AVD, а потом через некоторое время в AVD появляется экран блокировки.

Когда вы снимете блокировку с экрана AVD, махнув вверх на изображении замка, появляется только что созданное вами приложение. Имя приложения отображается в верхней части экрана, а текст по умолчанию "Hello world!" выводится в основной области.

Подготовка среды
Построение приложения
Запуск приложения
Изменение приложения

Что же только что произошло?

Разобьем то, что происходит при запуске приложения, на несколько этапов:

1. **Android Studio запускает эмулятор, загружает AVD и устанавливает приложение.**

2. **Когда приложение запустится, на базе MainActivity.java создается объект активности.**

3. **Активность указывает, что она использует макет activity_main.xml.**

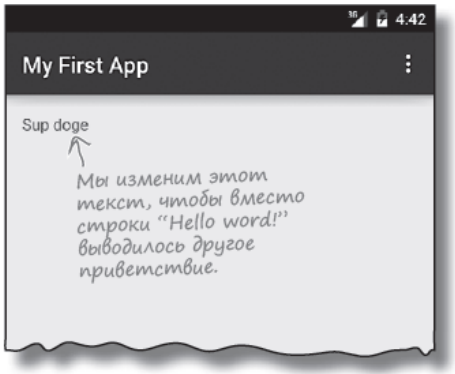
4. **Активность приказывает Android вывести макет на экран.** Текст “Hello world!” появляется на экране.

Подготовка среды
Построение приложения
Запуск приложения
Изменение приложения

Модификация приложения

На нескольких последних страницах мы создали простейшее Android-приложение и увидели, как оно выполняется в эмуляторе. Теперь мы займемся усовершенствованием только что созданного приложения.

Сейчас приложение выводит стандартный текст "Hello world!", включенный в него мастером. Мы заменим этот текст, чтобы приложение приветствовало пользователя как-то иначе. Что же для этого нужно сделать? Чтобы получить ответ на этот вопрос, отступим на шаг назад и посмотрим, как в настоящее время строится приложение.



Приложение состоит из одной активности и одного макета

Во время построения приложения мы сообщили Android Studio, как следует настроить его, а мастер сделал все остальное. Мастер сгенерировал базовую активность, а также макет по умолчанию.

Активность управляет тем, что делает приложение

Android Studio создает за нас активность с именем *MainActivity.java*. Активность определяет, что приложение **делает** и как оно должно реагировать на действия пользователя.

Макет управляет внешним видом приложения

Файл *MainActivity.java* указывает, что он использует макет с именем *activity_main.xml*, который среда Android Studio сгенерировала за нас. Макет определяет, как должно **выглядеть** приложение.

Итак, мы собираемся изменить внешний вид приложения, изменяя выводимый им текст. Это означает, что нам придется иметь дело с компонентом Android, управляющим внешним видом приложения. Значит, нужно поближе познакомиться с *макетом*.

Что содержит макет?

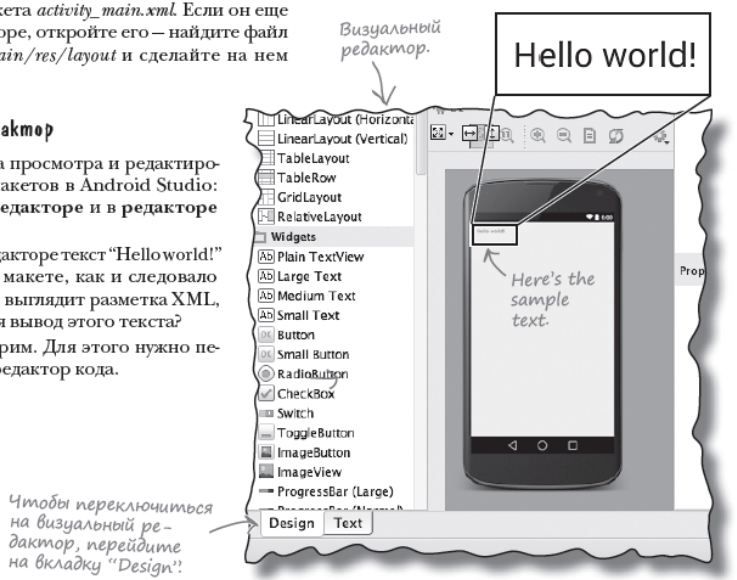
Мы собираемся изменить текст “Hello world!”, который среда Android Studio создала за нас, поэтому начнем с файла макета *activity_main.xml*. Если он еще не открыт в редакторе, откройте его — найдите файл в папке *app/src/main/res/layout* и сделайте на нем двойной щелчок.

Визуальный редактор

Есть два способа просмотра и редактирования файлов макетов в Android Studio: в **визуальном редакторе** и в **редакторе кода**.

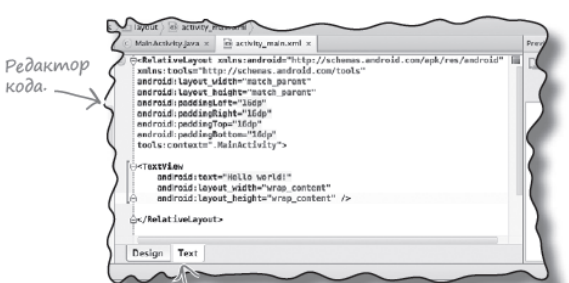
В визуальном редакторе текст “Hello world!” отображается в макете, как и следовало ожидать. Но как выглядит разметка XML, обеспечивающая вывод этого текста?

Давайте посмотрим. Для этого нужно переключиться в редактор кода.

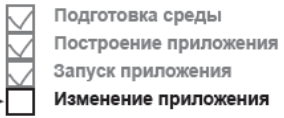


Редактор кода

Если выбрать редактор кода, в окне отображается содержимое файла *activity_main.xml*. Присмотримся к нему внимательнее.



activity_main.xml состоит из двух элементов



Ниже приведен код из файла *activity_main.xml*, сгенерированного Android Studio.

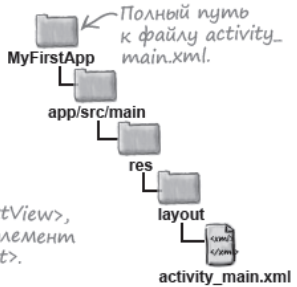
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    ...
    tools:context=".MainActivity" >

    <TextView
        android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

Это элемент <Relative Layout>.

Здесь идет разметка XML, сгенерированная Android Studio, но для нас она пока интереса не представляет.

Элемент <TextView>, вложенный в элемент <RelativeLayout>.



Код состоит из двух элементов.

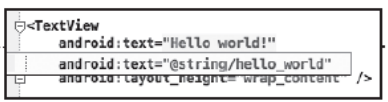
Первый элемент — `<RelativeLayout>` — приказывает Android выводить компоненты макета в относительных позициях. Например, элемент `<RelativeLayout>` может использоваться для выравнивания компонентов по центру макета, выравнивания по нижнему краю экрана Android-устройства или позиционирования относительно других компонентов.

Второй элемент — `<TextView>` — используется для вывода текста. Он вложен в элемент `<RelativeLayout>` и в нашем примере используется для вывода сгенерированного текста "Hello world!".

Ключевая часть кода в элементе `<TextView>` находится в первой строке. Вы ничего не замечаете?

Android Studio иногда отображает значения ссылок вместо реального кода.

Например, вместо реального кода `"@string/hello_world"` *может отображаться текст* «`Hello world!`». *Все такие подстановки должны выделяться цветом в редакторе кода; если щелкнуть на них или навести указатель мыши, то откроется настоящий код.*




```
<TextView
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Элемент TextView описывает текст в макете.

Ничего не заметили в этой строке?

Файл макета содержит ссылку на строку, а не саму строку

Ключевой частью элемента `<TextView>` является первая строка:

```
android:text="@string/hello_world" />
```

Запись `android:text` означает, что речь идет о свойстве `text` элемента `<TextView>`, а конструкция определяет, какой текст должен выводиться в макете. Но почему используется синтаксис `"@string/hello world"` вместо простого "Hello world!"? И что это вообще значит?

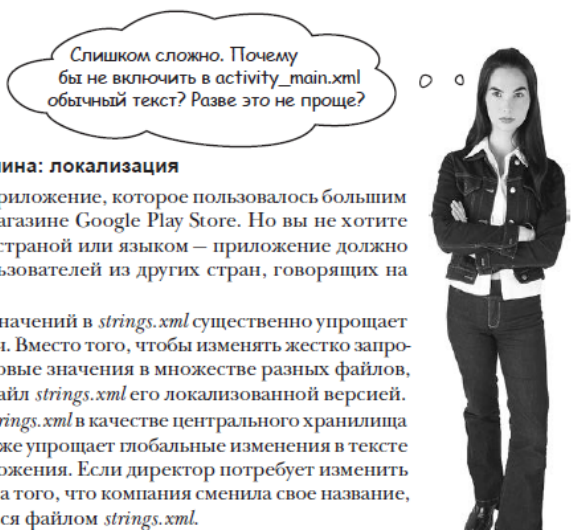
Начнем с первой части, `@string`. Она просто приказывает Android найти текстовое значение в файле строковых ресурсов. Внашем примере Android Studio создает файл строковых ресурсов с именем *strings.xml*, который находится в папке *app/src/main/res/values*. Вторая часть, `hello_world`, приказывает Android **получить значение ресурса с именем `hello_world`**. Таким образом, `@string/hello_world` означает: "Найти строковый ресурс с именем `hello_world` и использовать связанное с ним текстовое значение".

Размещайте строковые значения в strings.xml вместо того, чтобы жестко программировать их. strings.xml — файл ресурсов, используемый для хранения пар «имя / значение строки». Макеты и активности могут обращаться к строковым значениям по имени.

Вывести текст...

...строкового ресурса hello_world.

```
android:text="@string/hello_world" />
```



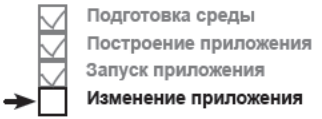
Одна важнейшая причина: локализация

Допустим, вы создали приложение, которое пользовалось большим успехом в локальном магазине Google Play Store. Но вы не хотите ограничиваться одной страной или языком — приложение должно быть доступно для пользователей из других стран, говорящих на других языках.

Выделение текстовых значений в *strings.xml* существенно упрощает решение подобных задач. Вместо того, чтобы изменять жестко запрограммированные текстовые значения в множестве разных файлов, достаточно заменить файл *strings.xml* его локализованной версией.

Использование файла *strings.xml* в качестве центрального хранилища текстовых значений также упрощает глобальные изменения в тексте в масштабах всего приложения. Если директор потребует изменить текст в приложении из-за того, что компания сменила свое название, достаточно ограничиться файлом *strings.xml*.

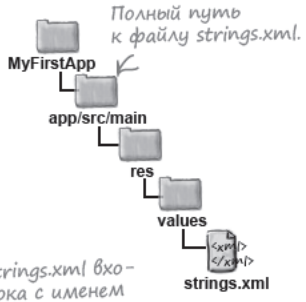
Заглянем в файл strings.xml



Среда Android Studio автоматически создала файл строковых ресурсов с именем *strings.xml*; давайте посмотрим, содержит ли этот файл ресурс `hello_world`. Найдите его в папке *app/src/main/res/values* на панели структуры проекта и откройте двойным щелчком.

Вот как выглядит код в *strings.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```



Как видно из листинга, в файле присутствует строка, которая, похоже, нам и нужна. Она описывает строковый ресурс с именем `hello_world` и значением "Hello world!":

```
<string name="hello_world">Hello world!</string>
```

Отредактируйте файл strings.xml, чтобы изменить выводимый текст

Итак, давайте изменим текст, выводимый приложением. Если вы этого еще не сделали ранее, найдите файл *strings.xml* на панели структуры проекта в Android Studio и сделайте на нем двойной щелчок, чтобы открыть его.

Ниже приведен код из файла. Найдите строку с именем "hello_world" и замените текстовое значение "Hello world!" другим – например, "Sup doge":

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_world">Hello world! Sup doge</string>
    <string name="action_settings">Settings</string>
</resources>
```

Замените значение "Hello world!" на "Sup doge".

После изменения файла перейдите в меню File и выберите команду Save All, чтобы сохранить изменения.

strings.xml — файл ресурсов по умолчанию. В этом файле хранятся строки в виде пар «имя/значение» для последующих обращений к ним из приложения. Строковые ресурсы имеют следующий формат:

Элемент <string> определяет пару «имя/значение» как строковый ресурс.

Элемент <resources> идентифицирует содержимое файла как ресурсы.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Существуют два признака, по которым Android распознает *strings.xml* как файл строковых ресурсов:

- **Файл хранится в папке app/src/main/res/values.** Файлы XML, которые хранятся в этой папке, содержат простые значения — строки, определения цветов и т.д.
- **Файл содержит элемент <resources>, который в свою очередь содержит один или несколько элементов <string>.** Из формата самого файла следует, что он представляет собой файл ресурсов для хранения строк. Элемент `<resources>` сообщает Android, что файл содержит ресурсы, а элемент `<string>` идентифицирует каждый строковый ресурс.

Это означает, что файл строковых ресурсов не обязан называться *strings.xml*; ему можно присвоить другое имя или разбить строковые ресурсы по нескольким файлам.
Каждая пара «имя/значение» имеет формат

```
<string name="имя_строки">значение_строки</string>
```

где `имя_строки` — идентификатор строки, а `значение_строки` — собственно строковое значение.
Для обращения к значению строки из макета используется синтаксис вида

```
"@string/имя_строки"
```

Имя строки, значение которой требуется получить.

Префикс "string" приказывает Android искать строковый ресурс с заданным именем.

Тест-драйв

- Подготовка среды
- Построение приложения
- Запуск приложения
- **Изменение приложения**

После того как файл будет отредактирован, попробуйте снова запустить приложение в эмуляторе — выберите команду "Run 'app'" из меню Run. На этот раз приложение должно вывести сообщение "Sup doge" вместо "Hello world!".

Обновленная версия приложения в эмуляторе.

ВИСНОВКИ

- Версии Android характеризуются номером версии, уровнем API и кодовым именем.
- Android Studio — специализированная версия среды IntelliJ IDEA, интегрированная с пакетом Android Software Development Kit (SDK) и системой сборки gradle.
- Типичное Android-приложение состоит из активностей, макетов и файлов ресурсов.
- Макеты описывают внешний вид приложения. Они хранятся в папке *app/src/main/res/layout*.
- Активности описывают то, что делает приложение, и как оно взаимодействует с пользователем. Созданные вами активности хранятся в папке *app/src/main/java*.
- Файл *strings.xml* содержит пары «имя/значение» для строк. Они используются для вынесения конкретных текстовых значений из макетов и активностей, а также для поддержки локализации.
- Файл *AndroidManifest.xml* содержит информацию о самом приложении. Этот файл находится в папке *app/src/main*.
- AVD — виртуальное устройство Android (Android Virtual Device). AVD выполняется в эмуляторе Android и моделирует физическое устройство Android.
- APK — пакет приложения Android, аналог JAR-файла для приложений Android. Файл содержит байт-код приложения, библиотеки и ресурсы. Установка приложения на устройстве осуществляется установкой его пакета APK.
- Приложения Android выполняются в отдельных процессах с использованием исполнительной среды Android (ART).
- Элемент `RelativeLayout` используется для размещения компонентов графического интерфейса в относительных позициях в макете.
- Элемент `TextView` используется для вывода текста.