разработка под Android SDK

В этом руководстве вы усвоите, как начать разработку под Андроид, а именно: мобильных приложений на платформе Android Eclipse + SDK. Сегодня существует множество подходов к разработке и созданию мобильных приложений для Android. Сложные аппаратные устройства, планшетные ПК и различные программные платформы (Symbian OS, iOS, WebOS, Windows Phone 7…) открывают бескрайнее пространство для разработчиков.

Итак, список приложений для начала разработки мобильных приложений под Android огромен. Какую платформу выбрать? Какой язык программирования учить? Какой набор приложений выбрать для планирования проекта? В этом руководстве вы усвоите, как начать разработку мобильных приложений для платформы Android, открытую мобильную ОС от Google.

Почему платформа Android?

Android - открытая платформа, основанная на ядре Linux. Установлена она на тысячах девайсов широкого круга производителей. Android предоставляет приложения для всех видов оборудования, которые только можно представить в виде современных мобильных устройств - цифровые компасы, видеокамеры, GPS, датчики и многое другое.

Бесплатные инструменты разработки для Android позволяют быстро начать создание бесплатных или почти бесплатных приложений. Когда вы готовы явить миру вашу программку, вы можете опубликовать ее посредством Android Market. Публикация в Android Market требует одноразового регистрационного взноса (25 долл. США на момент написания статьи) и, в отличие от App Store Apple (который очень строго подходит к цензу), делает ваше приложение доступным для скачивания и покупки после быстрого обзора - если приложение не нарушает правила и закон.

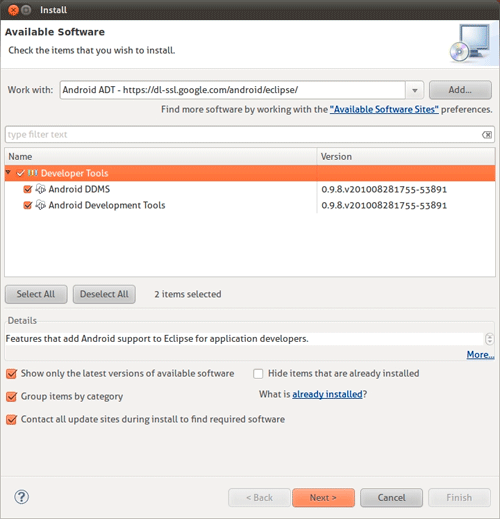
Вот несколько других отличий Android SDK, которые предлагает Вам преимущества как разработчику:

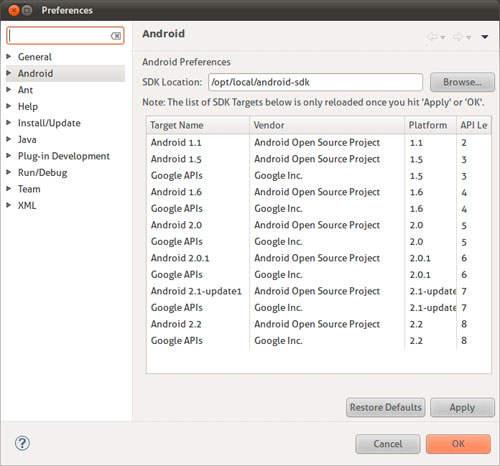
* Пакет Android SDK доступен для Windows, Mac и Linux, поэтому вам не нужно платить за новое "железо" для написания приложений.
* SDK встроен в Java. Если вы знакомы с языком программирования Java, вы уже на полпути к освоению.
* С учетом распространения приложения через Android Market, оно будет доступно сразу для сотен тысяч пользователей. Вы не ограничивается только официальным Market'ом, поскольку есть альтернативы. Например, вы можете опубликовать приложение на своем блоги. Amazon по слухам, готовит собственный магазин Android-приложений.
* Равно как техническая документация SDK, для разработчиков Android создаются новые ресурсы. Платформа набирает все большую популярность среди пользователей и разработчиков.

Хватит говорить - давайте начнем разработку приложений для Android!

Установка Eclipse и Android SDK

Рекомендуемое окружение для разработки Android приложений - это Eclipse с установленным плагином Android Development Toolkit (ADT). Я опишу все далее.



* Скачайте Android SDK для вашей платформы (Windows, Mac OS X или Linux).
* Распакуйте скачанные файлы в любое запоминающееся место на жестком диске (в Linux я придерживаюсь пути /opt/local/).
* Если Eclipse уже установлен, скачайте и установите пакет [Eclipse IDE for Java Developers](http://eclipse.org/downloads/packages/eclipse-ide-java-developers/galileosr2). Для программирования Google рекомендует использовать Eclipse 3.5 (Galileo).
* Запустите Eclipse и выберите Help->Install New Software.
* Нажмите Add в окне Available Software.
* Введите Android Development Tools в поле Name и https://dl-ssl.google.com/android/eclipse/ в поле Location.
* Нажмите OK и проверьте Developer Tools в списке доступного ПО. Этим самым мы установим Android Development Tools и DDMS, отладочный инструмент для Android.
* Нажмите Next и Finish для установки плагина. Нужно перезапустить Eclipse единожды после установки.
* После перезапуска Eclipse выберите Window->Preferences, и вы увидите Android в списке категорий.
* Теперь нужно указать Eclipse, где расположен установленный Android SDK. Нажмите Android, затем Browse для выбора местоположения распакованных файлов SDK. К примеру, /opt/local/android-sdk.  
    
  
* Нажмите OK для запоминания Eclipse'ом расположения SDK.

Ориентация под платформу Android при разработке

Перед тем, как начать написание приложений для Android, нужно скачать SDK для платформы Android. Каждая платформа имеет собственную версию Android SDK, которая установлена на устройствах пользователей. Для версии Android 1.5 и выше существует две платформы: Android Open Source Project и Google.

Android Open Source Project - платформа с открытым исходным кодом, но не включает в себя расширения Google, такие как Google Maps. Если вы не хотите использовать Google API, функциональность карт Google будет недоступна для вашего приложения. Если у вас нет особых причин делать это, я бы рекомендовал вам настроить таргетинг на одной из платформ Google, так как это позволит вам использовать собственные расширения Google.

* Выберите Window->Android SDK and AVD Manager.
* Нажмите Available Packagesв левой колонке и проверьте репозиторий, чтобы показать список доступных платформ Android.
* Вы можете выбрать платформы для загрузки из списка или оставить все отмеченными, чтобы загрузить все доступные платформы. По завершению, нажмите Install Selected и следуйте инструкции по установке.

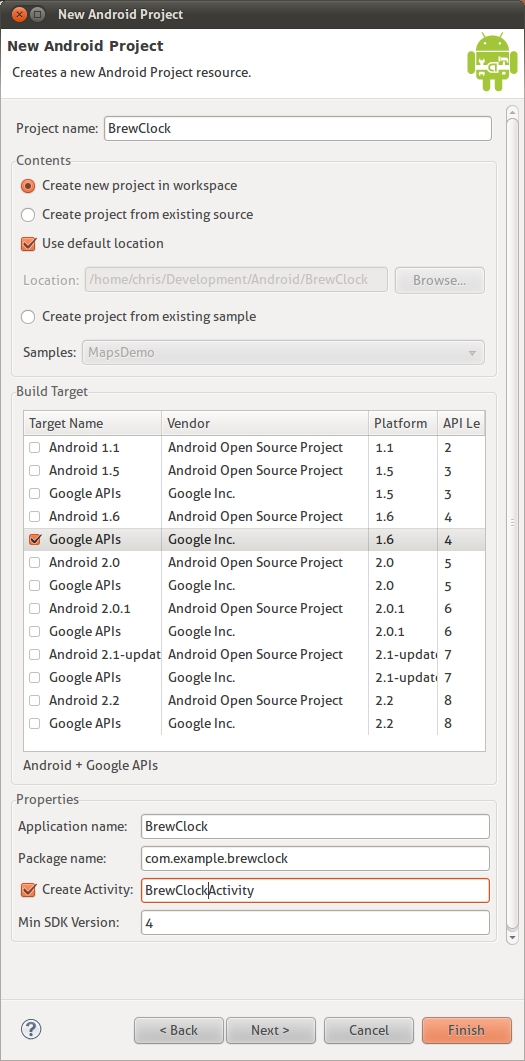
После того как все было успешно загружено, вы готовы приступить к разработке под Android.

Создание нового проекта Android

Мастер настройки Eclipse New Project Wizard поможет создать приложение Android, сгенерировав файлы и код, готовые к запуску, справа от блока. Это быстрый способ убедиться в работоспособности и хорошая начальная точка при начале разработки приложений:

* Выберите File->New->Project…
* Выберите Android Project
* В диалоге New Project введите следующие настройки:
* Project Name: BrewClock
* Build Target: Google Inc. 1.6 (Api Level 4)
* Application Name: BrewClock
* Package Name: com.example.brewclock
* Create Activity: BrewClockActivity

Min SDK Version: 4



После нажатия Finish, Eclipse создаст новый проект Android, готовый к запуску и разработке под Android. Помните, вы говорили Eclipse сгенерировать Activity, названный BrewClockActivity? Это код, который Android на самом деле использует для запуска приложения. Сгенерированный код будет отображен в виде простого сообщения в духе ‘Hello World’.

Пакеты

Имя пакета - идентификатор для вашего приложения. Когда настанет время публикации результата на Android Market, этот идентификатор будет использоваться для отслеживания обновлений для приложения, поэтому важно убедиться, что этот идентификатор уникален. Хотя здесь мы используем имя com.example.brewclock,для реального приложения лучше выбрать что-то вроде com.yourcompanyname.yourapplication.

Версии SDK

Версия Min SDK Version (название говорит само за себя) - самая ранняя версия Android, на которой приложение будет работать. С каждым новым выпуском Android, SDK добавляет и изменяет различные методы. При выборе версии SDK, Android (и Android Market) знает, что ваше приложение будет работать только на устройствах с версией платформы Android указанной версии и выше.

Запуск вашего приложения

Сейчас попробуем запустить наше приложение в Eclipse. Поскольку это первый запуск, Eclipse спросит, с каким из типов проекта вы работаете:

* Выберите Run->Run или нажмите Ctrl+F11.
* Выберите Android Application и нажмите OK.

Eclipse попробует запустить приложение на устройстве с Android. На данный момент, однако, у вас нет устройств под управлением Android, так что запустить проект не удастся, и вам будет предложено создать новое устройство Android Virtual Device (AVD).

Виртуальные устройства (Android Virtual Devices)

Android Virtual Device (AVD) эмулирует среду Android, будь то мобильный телефон или планшетный ПК. Вы можете создавать сколько угодно устройств AVD, с разными версиями платформы Android. Для каждого AVD вы можете настроить различные параметры устройств, указав наличие физической клавиатуры, поддержки GP, разрешение камеры и так далее.

Перед тем как запустить приложение, вы должны создать свой первое устройство AVD с платформой SDK (Google API, 1,6).

Давайте сделаем это сейчас:

* Если вы еще не пробовали запустить приложение, нажмите Run (или сочетание клавиш Ctrl+F11)
* Когда всплывает предупреждение, нажмите Yes для создания AVD.
* Нажмите New в диалоге Android SDK and AVD Manager.
* Установите следующие параметры для AVD:
* Name: Android\_1.6
* Target: Google APIs (Google Inc.) - API Level 4
* SD Card Size: 16 MiB

Skin Built In: Default (HVGA)

* Нажмите Create AVD.
* Закройте диалог Android SDK and AVD Manager.

Запуск кода

Попробуйте запустить приложение снова (Ctrl + F11). Eclipse теперь будет создавать ваш проект и запускать новый AVD. Помните, AVD полностью эмулирует среду Android, причем вам даже не нужно наблюдать достаточно медленный процесс загрузки, как в случае с реальным устройством. По этой причине, как только AVD будет готов к работе, лучше не закрывать окно, пока вы закончили программировать, в течение всего дня.

Когда [android-эмулятор](http://softdroid.net/luchshie-besplatnye-emulyatory-dlya-os-android) загрузится, Eclipse автоматически установит и запустит приложение:



Создание вашего первого приложения на Android

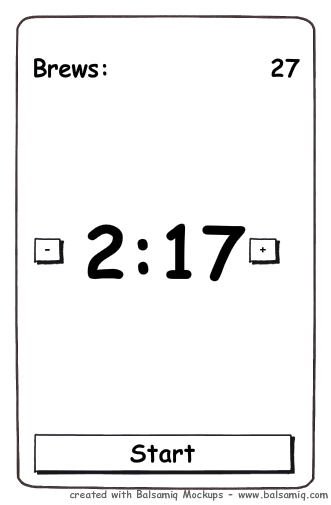
Тестирование сгенерированного кода - это хорошо, но вы хотите начать создание реальной программы. Для этого, мы будем двигаться от простого процесса разработки и создания приложений, которое можно развернуть на устройстве Android.

Большинство разработчиков (включая меня), постоянное снабжают себя хорошим чаем или кофе. В следующем разделе этой статьи вы будете создавать счетчик приложение для отслеживания, сколько чашек чая (пива) пользователь опустошил и позволит установить таймер для варки каждой чашки.

Вы можете скачать полный код для этой обучающей программы на [GitHub](http://github.com/cblunt/brewclock).

Разработка под андроид: проектирование пользовательского интерфейса

Одним из первых шагов к созданию какой-либо программы вообще - проектирование пользовательского интерфейса. Вот небольшой набросок нашего приложения:

  
[Увеличить изображение](http://media.smashingmagazine.com/cdn_smash/wp-content/uploads/2010/10/design_sketch.jpg)

Пользователь сможет установить время кипячения в минутах, используя кнопки + и -. Когда он нажмет Start, отсчет времени начнется в течение указанного количества минут. Если пользователь отменяет кипячение, нажав на кнопку еще раз, время будет увеличено, когда таймер достигнет 0.

Построение интерфейса

Пользовательский интерфейс Android, или оболочка, которые описаны в документации XML, могут быть найдены в папке res/layouts. Код шаблона, уже сгенерированный Eclipse, объявлен в res/layouts/main.xml и, как вы заметили, уже работает в эмуляторе.

Eclipse располагает собственным инструментарием для проектировки макета, который позволяет создавать интерфейс методом перетаскивания в пределах экрана. Тем не менее, временами легче написать интерфейс в XML и использовать графический макет для предварительного просмотра результатов.

Давайте сделаем это сейчас, изменив main.xml в соответствии с эскизом выше:

* Откройте res/layouts/main.xml в Eclipse двойным нажатием из Package Explorer.
* Выберите вкладку main.xml в нижней части экрана, чтобы перейти в режим XML.

Сейчас изменим содержание main.xml на следующее:

# /res/layouts/main.xml

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

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

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

<LinearLayout

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:padding="10dip">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="20dip"

android:text="Brews: " />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="None"

android:gravity="right"

android:textSize="20dip"

android:id="@+id/brew\_count\_label" />

</LinearLayout>

<LinearLayout

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:gravity="center"

android:padding="10dip">

<Button

android:id="@+id/brew\_time\_down"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="-"

android:textSize="40dip" />

<TextView

android:id="@+id/brew\_time"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="0:00"

android:textSize="40dip"

android:padding="10dip" />

<Button

android:id="@+id/brew\_time\_up"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="+"

android:textSize="40dip" />

</LinearLayout>

<Button

android:id="@+id/brew\_start"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom"

android:text="Start" />

</LinearLayout>

Как вы можете видеть, файлов в Android XML много, но это позволит вам контролировать практически каждый элемент на экране.

Одним из наиболее важных элементов интерфейса в Android являются контейнеры Layout, такие как LinearLayout, используемый в этом примере. Эти элементы невидимы для пользователя, но выступают в качестве контейнеров для других элементов, таких как Buttons и TextViews.

Есть несколько типов расположения макета, каждый из которых используется для построения различных типов планировки. Как и LinearLayout и AbsoluteLayout, TableLayout позволяет использование интерфейса с использованием сетки. Вы можете узнать больше об этом в общий макет объектов разделе API-документации.

Привязка вашего макета к коду

После сохранения макет, попробуйте запустить приложение в эмуляторе, нажав Ctrl + F11, или нажмите значок Run в Eclipse. Теперь вместо сообщения "Hello World" вы увидите, что в Android теперь отображается интерфейс приложения.

Если нажать любую кнопку, они будут подсвечиваться, как и ожидалось, но не делайте что-нибудь помимо этого. Давайте продолжим написание кода вслед за интерфейсом макета:

# /src/com/example/brewclock/BrewClockActivity.java

...

import android.widget.Button;

import android.widget.TextView;

public class BrewClockActivity extends Activity {

/\*\* Properties \*\*/

protected Button brewAddTime;

protected Button brewDecreaseTime;

protected Button startBrew;

protected TextView brewCountLabel;

protected TextView brewTimeLabel;

...

}

Далее, мы изменим вызов onCreate. Это вызов происходит всякий раз, когда на Android запускается приложение. В коде, сгенерированном Eclipse, onCreateсоздает вид активности на R.layout.main. Это та строка кода, которая дает команду Android декодировать макет документа XML и отображать его пользователю.

Объект ресурса (Resource Object)

В Android, R - это специальный объект, который автоматически создается для обеспечения доступа к ресурсам вашего проекта (макеты, строки, меню, иконки...) внутри кода. Каждому ресурсу присваивается свой id. В файле макета (см. выше) это @+id атрибуты XML. Мы будем использовать их для привязки Buttons и TextViews в нашем макете:

# /src/com/example/brewclock/BrewClockActivity.java

...

public class BrewClockActivity extends Activity {

...

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

// Connect interface elements to properties

brewAddTime = (Button) findViewById(R.id.brew\_time\_up);

brewDecreaseTime = (Button) findViewById(R.id.brew\_time\_down);

startBrew = (Button) findViewById(R.id.brew\_start);

brewCountLabel = (TextView) findViewById(R.id.brew\_count\_label);

brewTimeLabel = (TextView) findViewById(R.id.brew\_time);

}

}

"Прослушка" событий

Для определения нажатия кнопок нам необходимо реализовать их прослушивание. Вы можете быть знакомы со слушателями обратного вызова, или callbacks, из других платформ, таких как Javascript / JQuery или Rails.

Android обеспечивает аналогичный механизм путем предоставления интерфейса Listener, такие как OnClickListener, которые определяют методы, которые будут срабатывать при возникновении события. OnClickListener интерфейс уведомит ваше приложение, когда пользователь нажимает на экран или определенную кнопку. Кроме того, необходимо сказать каждой кнопке о ClickListener:

# /src/com/example/brewclock/BrewClockActivity.java

...

// Be sure not to import

// `android.content.dialoginterface.OnClickListener`.

import android.view.View.OnClickListener;

public class BrewClockActivity extends Activity

implements OnClickListener {

...

public void onCreate(Bundle savedInstanceState) {

...

// Setup ClickListeners

brewAddTime.setOnClickListener(this);

brewDecreaseTime.setOnClickListener(this);

startBrew.setOnClickListener(this);

}

...

public void onClick(View v) {

// TODO: Add code to handle button taps

}

}

Далее мы добавим код, который обрабатывает каждое из наших нажатий кнопок. Мы также добавим четыре новые свойства Activity, что позволит пользователю установить и отслеживать время пивоварения: сколько варок было сделано и запущен ли таймер.

# /src/com/example/brewclock/BrewClockActivity.java

...

public class BrewClockActivity extends Activity

implements OnClickListener {

...

protected int brewTime = 3;

protected CountDownTimer brewCountDownTimer;

protected int brewCount = 0;

protected boolean isBrewing = false;

...

public void onClick(View v) {

if(v == brewAddTime)

setBrewTime(brewTime + 1);

else if(v == brewDecreaseTime)

setBrewTime(brewTime -1);

else if(v == startBrew) {

if(isBrewing)

stopBrew();

else

startBrew();

}

}

}

Обратите внимание, что мы используем CountDownTimer класс, предоставляемый Android. Это позволяет легко создать и запустить простой счетчик и получать уведомления через регулярные промежутки времени в то время, как идет обратный отсчет. Вы будете использовать это в методе startBrew, читайте об этом ниже.

Следующие методы - модели поведения ползунка настройки времени варки, запуска и остановки варения, а также счетчика. Мы также инициализируем свойства brewTime и brewCount в onCreate.

Было бы хорошей практикой переместить этот код в отдельный класс модели, но для простоты мы добавим код в BrewClockActivity:

# /src/com/example/brewclock/BrewClockActivity.java

...

public class BrewClockActivity extends Activity

implements OnClickListener {

...

public void onCreate(Bundle savedInstanceState) {

...

// Set the initial brew values

setBrewCount(0);

setBrewTime(3);

}

/\*\*

\* Set an absolute value for the number of minutes to brew.

\* Has no effect if a brew is currently running.

\* @param minutes The number of minutes to brew.

\*/

public void setBrewTime(int minutes) {

if(isBrewing)

return;

brewTime = minutes;

if(brewTime < 1)

brewTime = 1;

brewTimeLabel.setText(String.valueOf(brewTime) + "m");

}

/\*\*

\* Set the number of brews that have been made, and update

\* the interface.

\* @param count The new number of brews

\*/

public void setBrewCount(int count) {

brewCount = count;

brewCountLabel.setText(String.valueOf(brewCount));

}

/\*\*

\* Start the brew timer

\*/

public void startBrew() {

// Create a new CountDownTimer to track the brew time

brewCountDownTimer = new CountDownTimer(brewTime \* 60 \* 1000, 1000) {

@Override

public void onTick(long millisUntilFinished) {

brewTimeLabel.setText(String.valueOf(millisUntilFinished / 1000) + "s");

}

@Override

public void onFinish() {

isBrewing = false;

setBrewCount(brewCount + 1);

brewTimeLabel.setText("Brew Up!");

startBrew.setText("Start");

}

};

brewCountDownTimer.start();

startBrew.setText("Stop");

isBrewing = true;

}

/\*\*

\* Stop the brew timer

\*/

public void stopBrew() {

if(brewCountDownTimer != null)

brewCountDownTimer.cancel();

isBrewing = false;

startBrew.setText("Start");

}

...

}

Только части этого кода, характерные для Android, создают отображения меток, используя метод setText. В startBrew мы создаем и запускаем CountDownTimer для начала посекундного отсчета, когда варка/кипячение окончео. Обратите внимание, что мы определяем слушателей CountDownTimer ( onTick и onFinish). onTick будет вызываться каждые 1000 миллисекунд (1 секунду), в то время как вызов onFinish происходит тогда, когда таймер достигнет нуля.

Избегаем сложности в программировании на Android

Чтобы сохранить это руководство для разработки под Android простым, я намеренно применял надписи непосредственно в коде (например, "Brew Up!", "Start", "Stop"). На самом деле, это не очень хорошая практика, так как затрудняет поиск и изменение этих строк в крупных проектах.

Android обеспечивает аккуратный способ хранения ваших текстовых строк отдельно от кода R-объекта. R позволяет определить все строки приложения в XML-файле (res/values/strings.xml), к которому можно получить доступ в коде по ссылке. Например:

# /res/values/strings.xml

<string name="brew\_up\_label">Brew Up!</string>

...

# /res/com/example/brewclock/BrewClockActivity.java

...

brewLabel.setText(R.string.brew\_up\_label);

...

Теперь, если вы хотите изменить Brew Up! что-то другое, нужно изменить его единожды в файле strings.xml. Если ваше приложение содержит десятки файлов кода, надписей в одном месте приобретает огромный смысл!

Проверяем BrewClock

Мы завершили написание кода, и пришло время "опробовать" приложение. Нажмите "Run" или Ctrl + F11, чтобы запустить BrewClock в эмуляторе. Если все прошло хорошо, вы увидите интерфейс настройки и готовы к заварке чая! Попробуйте установить разное время заварки и нажмите Start для обратного отсчета.

Резюме

В этом кратком введении в разработку под Андроид, вы скачали и установили Android SDK и плагин Eclipse Android Development Tools (ADT). Вы настроили эмулятор или виртуальное устройство для тестирования ваших приложений. Вы также создали рабочее приложение для Android который подчеркнул ряд ключевых понятий, которые вы будете использовать при разработке будущих Android приложений.

Будем надеяться, что это разбудит ваш аппетит для создания мобильных приложений и экспериментов в этой захватывающей области. Android предлагает отличный способ к написанию приложений для ряда нынешних и будущих мобильных устройств. Если вы уже написали рабочее приложение под эту платформу, не забудьте сообщить нам об этом в комментариях!