1

Getting Started with Android Programming

WHAT YOU WILL LEARN IN THIS CHAPTER

- What is Android?
- Android versions and its feature set
- The Android architecture
- The various Android devices on the market
- The Android Market application store
- How to obtain the tools and SDK for developing Android applications
- How to develop your first Android application

Welcome! The fact that you are holding this book in your hands (or are reading it on your latest mobile device) signifies that you are interested in learning how to write applications for the Android platform — and there's no better time to do this than now! The mobile application market is exploding, and recent market research shows that Android has overtaken iPhone to occupy the second position in the U.S. smartphone market. The first place honor currently goes to Research In Motion (RIM), with Apple's iPhone taking third place. By the time you read this, chances are good that Android may have become the number one smartphone platform in the U.S., and that you may even be reading this on one of the latest Android devices.

What propelled this relatively unknown operating system, which Google bought in 2005, to its popular status today? And what features does it offer? In this chapter you will learn what Android is, and what makes it so compelling to both developers and device manufacturers alike. You will also get started with developing your first Android application, and learn how to obtain all the necessary tools and set them up. By the end of this chapter, you will be equipped with the basic knowledge you need to explore more sophisticated techniques and tricks for developing your next killer Android application.

WHAT IS ANDROID?

Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team).

Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code. Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others. This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phenomenon of Apple's iPhone, a hugely successful product that revolutionized the smartphone industry. Such companies include Motorola and Sony Ericsson, which for many years have been developing their own mobile operating systems. When the iPhone was launched, many of these manufacturers had to scramble to find new ways of revitalizing their products. These manufacturers see Android as a solution — they will continue to design their own hardware and use Android as the operating system that powers it.

The main advantage of adopting Android is that it offers a unified approach to application development. Developers need only develop for Android, and their applications should be able to run on numerous different devices, as long as the devices are powered using Android. In the world of smartphones, applications are the most important part of the success chain. Device manufacturers therefore see Android as their best hope to challenge the onslaught of the iPhone, which already commands a large base of applications.

Android Versions

Android has gone through quite a number of updates since its first release. Table 1-1 shows the various versions of Android and their codenames.

TABLE 1-1: A Brief History of Android Versions

ANDROID VERSION	RELEASE DATE	CODENAME
1.1	9 February 2009	
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 October 2009	Eclair
2.2	20 May 2010	Froyo
2.3	6 December 2010	Gingerbread
3.0	Unconfirmed at the time of writing	Honeycomb

Features of Android

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

- **Storage** — Uses SQLite, a lightweight relational database, for data storage. Chapter 6 discusses data storage in more detail.
- **Connectivity** — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX. Chapter 8 discusses networking in more detail.
- **Messaging** — Supports both SMS and MMS. Chapter 8 discusses messaging in more detail.
- **Web browser** — Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- **Hardware support** — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
- **Multi-touch** — Supports multi-touch screens
- **Multi-tasking** — Supports multi-tasking applications
- **Flash support** — Android 2.3 supports Flash 10.1.
- **Tethering** — Supports sharing of Internet connections as a wired/wireless hotspot

Architecture of Android

In order to understand how Android works, take a look at Figure 1-1, which shows the various layers that make up the Android operating system (OS).

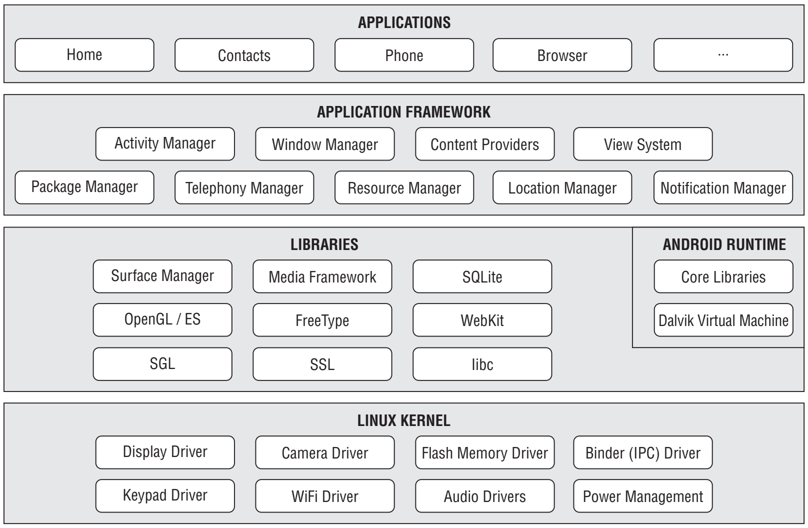


FIGURE 1-1

The Android OS is roughly divided into five sections in four main layers:

- **Linux kernel** — This is the kernel on which Android is based. This layer contains all the low-level device drivers for the various hardware components of an Android device.
- **Libraries** — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.
- **Android runtime** — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.
- **Application framework** — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.
- **Applications** — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

Android Devices in the Market

Android devices come in all shapes and sizes. As of late November 2010, the Android OS can be seen powering the following types of devices:

- Smartphones
- Tablets
- E-reader devices
- Netbooks
- MP4 players
- Internet TVs

Chances are good that you own at least one of the preceding devices. Figure 1-2 shows (clockwise) the Samsung Galaxy S, the HTC Desire HD, and the LG Optimus One smartphones.

FIGURE 1-2

Another popular category of devices that manufacturers are rushing out is the *tablet*. Tablet sizes typically start at seven inches, measured diagonally. Figure 1-3 shows the Samsung Galaxy Tab and the Dell Streak, which is a five-inch phone tablet.

Besides smartphones and tablets, Android is also beginning to appear in dedicated devices, such as e-book readers. Figure 1-4 shows the Barnes and Noble's NOOKcolor, which is a color e-Book reader running the Android OS.

FIGURE 1-3

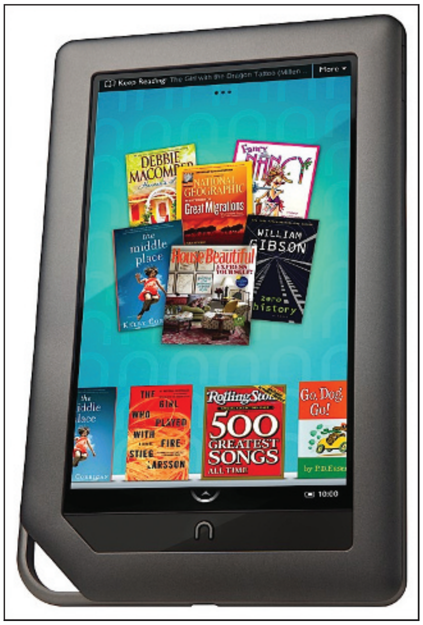


FIGURE 1-4

In addition to these popular mobile devices, Android is also slowly finding its way into your living room. People of Lava, a Swedish company, has developed an Android-based TV, call the Scandinavia Android TV (see Figure 1-5).

Google has also ventured into a proprietary smart TV platform based on Android and co-developed with companies such as Intel, Sony, and Logitech. Figure 1-6 shows Sony's Google TV.

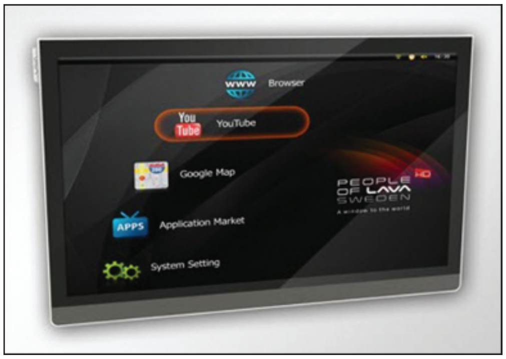


FIGURE 1-5

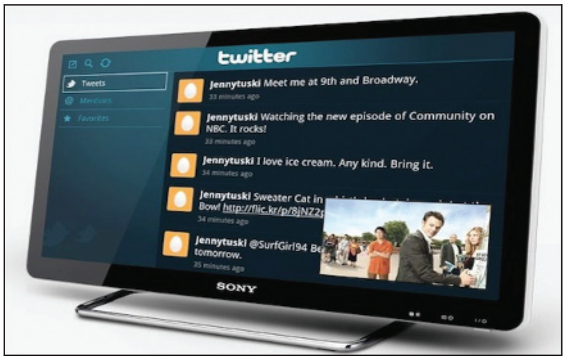


FIGURE 1-6

The Android Market

As mentioned earlier, one of the main factors determining the success of a smartphone platform is the applications that support it. It is clear from the success of the iPhone that applications play a very vital role in determining whether a new platform swims or sinks. In addition, making these applications accessible to the general user is extremely important.

As such, in August 2008, Google announced the Android Market, an online application store for Android devices, and made it available to users in October 2008. Using the Market application that is preinstalled on their Android device, users can simply download third-party applications directly onto their devices. Both paid and free applications are supported on the Android Market, though paid applications are available only to users in certain countries due to legal issues.

Similarly, in some countries, users can buy paid applications from the Android Market, but developers cannot sell in that country. As an example, at the time of writing, users in India can buy apps from the Android Market, but developers in India cannot sell apps on the Android Market. The reverse may also be true; for example, users in South Korea cannot buy apps, but developers in South Korea can sell apps on the Android Market.

Chapter 11 discusses more about the Android Market and how you can sell your own applications in it.

OBTAINING THE REQUIRED TOOLS

Now that you know what Android is and its feature set, you are probably anxious to get your hands dirty and start writing some applications! Before you write your first app, however, you need to download the required tools and SDKs.

For Android development, you can use a Mac, a Windows PC, or a Linux machine. All the tools needed are free and can be downloaded from the Web. Most of the examples provided in this book should work fine with the Android emulator, with the exception of a few examples that require access to the hardware. For this book, I will be using a Windows 7 computer to demonstrate all the code samples. If you are using a Mac or Linux computer, the screenshots should look similar; some minor differences may be present, but you should be able to follow along without problems.

So, let the fun begin!

JAVA JDK

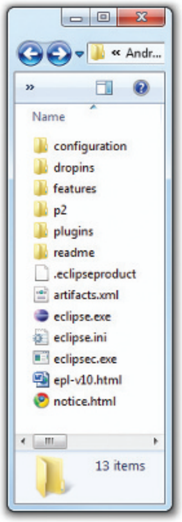
The Android SDK makes use of the Java SE Development Kit (JDK). Hence, if your computer does not have the JDK installed, you should start by downloading the JDK from `www.oracle.com/technetwork/java/javase/downloads/index.html` and installing it prior to moving to the next section.

Eclipse

The first step towards developing any applications is obtaining the integrated development environment (IDE). In the case of Android, the recommended IDE is Eclipse, a multi-language software development environment featuring an extensible plug-in system. It can be used to develop various types of applications, using languages such as Java, Ada, C, C++, COBOL, Python, etc.

For Android development, you should download the Eclipse IDE for Java EE Developers (`www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr1`). Six editions are available: Windows (32 and 64-bit), Mac OS X (Cocoa 32 and 64), and Linux (32 and 64-bit). Simply select the relevant one for your operating system. All the examples in this book were tested using the 32-bit version of Eclipse for Windows.

Once the Eclipse IDE is downloaded, unzip its content (the `eclipse` folder) into a folder, say `C:\Android\`. Figure 1-7 shows the content of the `eclipse` folder.



« Andr...

Name

configuration
dropins
features
p2
plugins
readme
.eclipseproduct
artifacts.xml
eclipse.exe
eclipse.ini
eclipsec.exe
epl-v10.html
notice.html

13 items

FIGURE 1-7

Android SDK

The next important piece of software you need to download is, of course, the Android SDK. The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.

You can download the Android SDK from `http://developer.android.com/sdk/index.html`.

Once the SDK is downloaded, unzip its content (the `android-sdk-windows` folder) into the `C:\Android\` folder, or whatever name you have given to the folder you just created.

Android Development Tools (ADT)

The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- Create new Android application projects.
- Access the tools for accessing your Android emulators and devices.
- Compile and debug Android applications.
- Export Android applications into Android Packages (APK).
- Create digital certificates for code-signing your APK.

To install the ADT, first launch Eclipse by double-clicking on the `eclipse.exe` file located in the `eclipse` folder.

When Eclipse is first started, you will be prompted for a folder to use as your workspace. In Eclipse, a workspace is a folder where you store all your projects. Take the default suggested and click OK.

Once Eclipse is up and running, select the Help ➪ Install New Software… menu item (see Figure 1-8).

In the Install window that appears, type **http://dl-ssl.google.com/android/eclipse** in the text box (see Figure 1-9) and click Add….

After a while, you will see the Developer Tools item appear in the middle of the window (see Figure 1-10). Expand it, and it will reveal its content: Android DDMS, Android Development Tools, and Android Hierarchy Viewer. Check all of them and click Next.

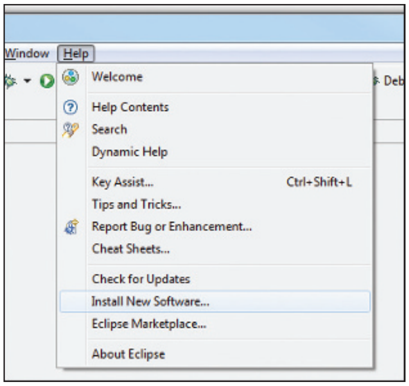


FIGURE 1-8

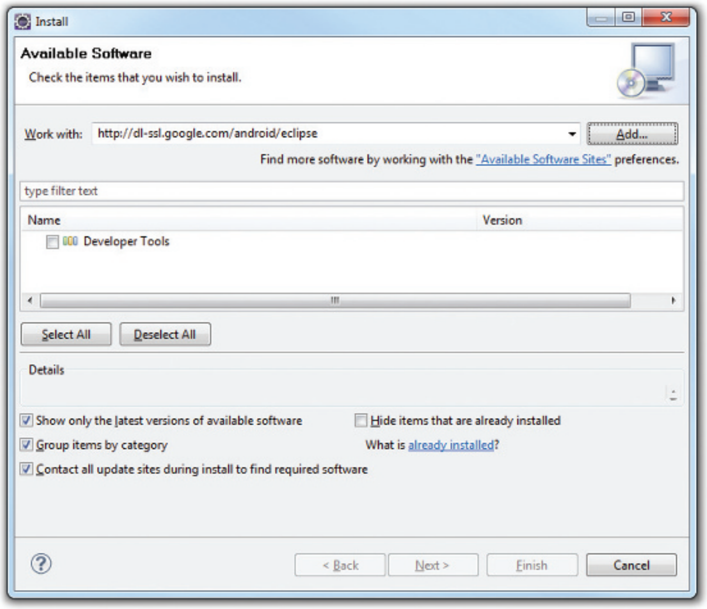
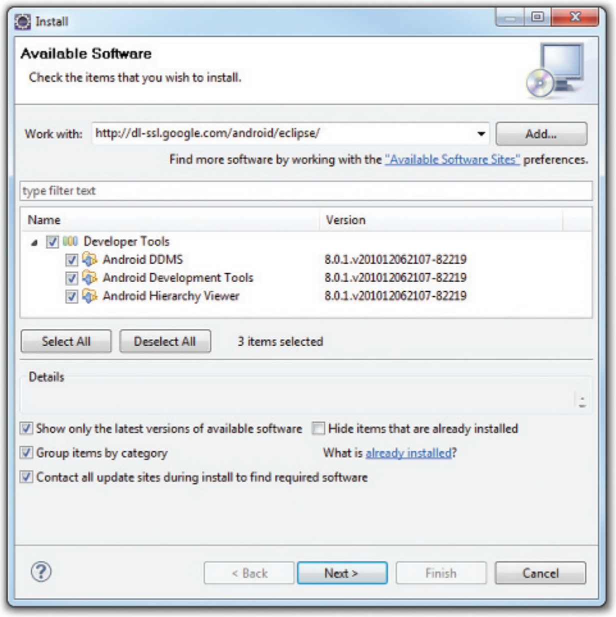

Install
Available Software
Check the items that you wish to install.
Work with: http://dl-ssl.google.com/android/eclipse
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
Developer Tools
Select All
Deselect All
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel

FIGURE 1-9


Install

Available Software

Check the items that you wish to install.

Work with: http://dl-ssl.google.com/android/eclipse/ Add...

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
[✓] Developer Tools	
[✓] Android DDMS	8.0.1.v201012062107-82219
[✓] Android Development Tools	8.0.1.v201012062107-82219
[✓] Android Hierarchy Viewer	8.0.1.v201012062107-82219

Select All Deselect All 3 items selected

Details

[✓] Show only the latest versions of available software [ ] Hide items that are already installed

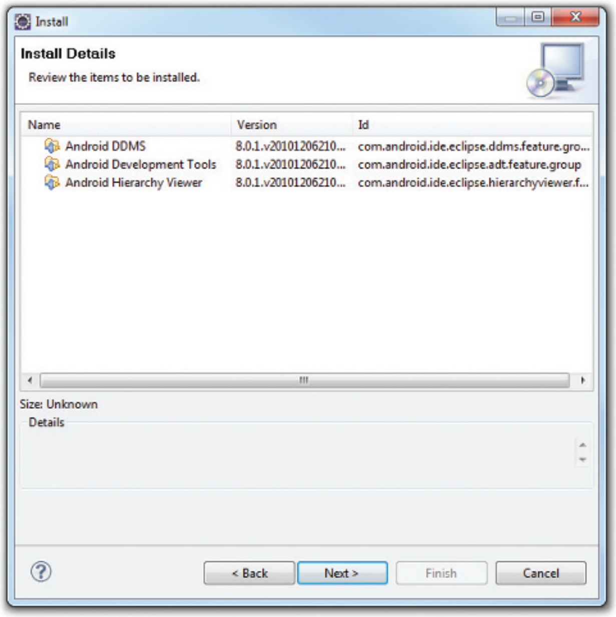
[✓] Group items by category What is already installed?

[✓] Contact all update sites during install to find required software

< Back Next > Finish Cancel

FIGURE 1-10

When you see the installation details, as shown in Figure 1-11, click Next.


Install

Install Details

Review the items to be installed.

Name	Version	Id
Android DDMS	8.0.1.v20101206210...	com.android.ide.eclipse.ddms.feature.gro...
Android Development Tools	8.0.1.v20101206210...	com.android.ide.eclipse.adt.feature.group
Android Hierarchy Viewer	8.0.1.v20101206210...	com.android.ide.eclipse.hierarchyviewer.f...

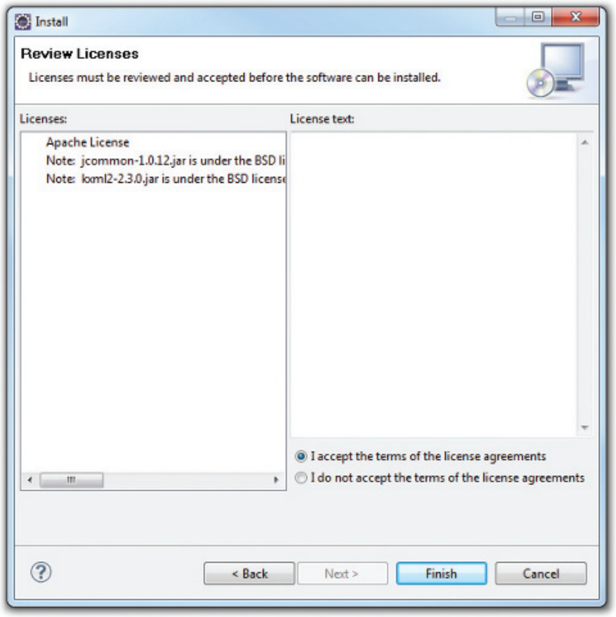
Size: Unknown

Details

< Back Next > Finish Cancel

FIGURE 1-11

You will be asked to review the licenses for the tools. Check the option to accept the license agreements (see Figure 1-12). Click Finish to continue.


Install
Review Licenses
Licenses must be reviewed and accepted before the software can be installed.
Licenses:
License text:
< Back
Next >
Finish
Cancel

Apache License
Note: jcommon-1.0.12.jar is under the BSD li
Note: kxml2-2.3.0.jar is under the BSD license

[✓ I accept the terms of the license agreements]
I do not accept the terms of the license agreements

FIGURE 1-12

Eclipse will now proceed to download the tools from the Internet and install them (see Figure 1-13). This will take some time, so be patient.

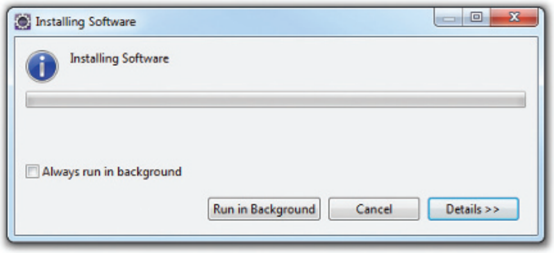


FIGURE 1-13

NOTE *If you have any problems downloading the ADT, check out Google's help at* `http://developer.android.com/sdk/eclipse-adt.html#installing`.

Once the ADT is installed, you will be prompted to restart Eclipse. After doing so, go to Window ➪ Preferences (see Figure 1-14).

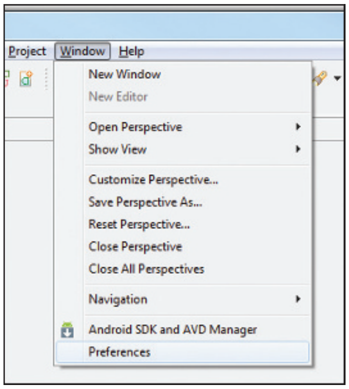
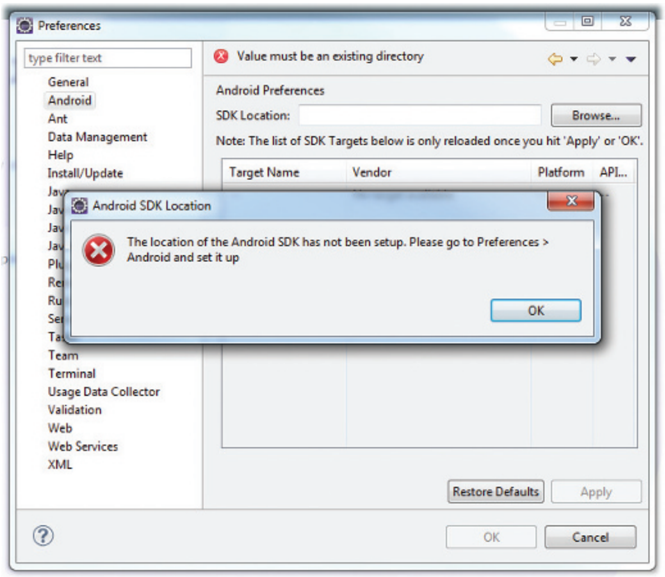


FIGURE 1-14

In the Preferences window that appears, select Android. You will see an error message saying that the SDK has not been set up (see Figure 1-15). Click OK to dismiss it.


Preferences

Value must be an existing directory

Android Preferences

SDK Location:

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API...

Android SDK Location

The location of the Android SDK has not been setup. Please go to Preferences > Android and set it up

FIGURE 1-15

Enter the location of the Android SDK folder. In this example, it would be `C:\Android\android-sdk-windows`. Click OK.

Creating Android Virtual Devices (AVDs)

The next step is to create AVD to be used for testing your Android applications. AVD stands for Android Virtual Devices. An AVD is an emulator instance that enables you to model an actual device.

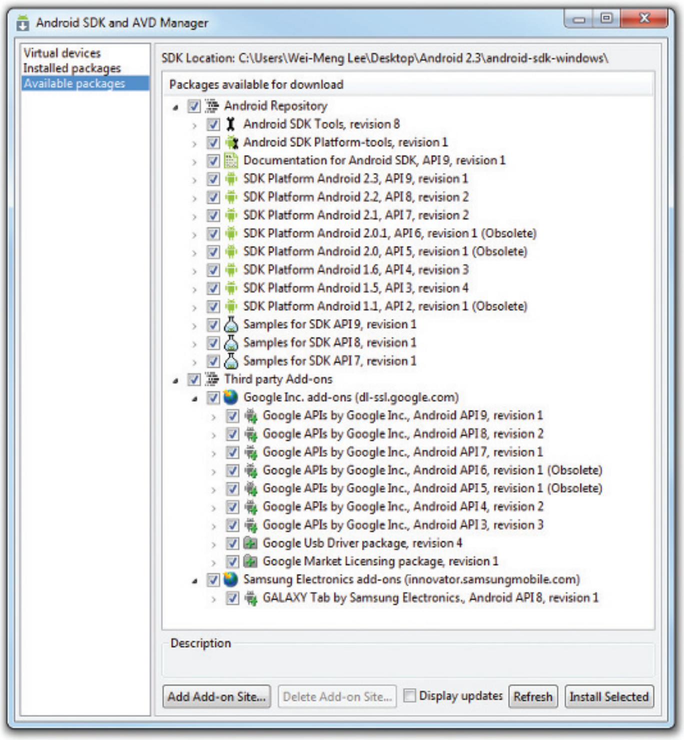
Each AVD consists of a hardware profile, a mapping to a system image, as well as emulated storage, such as a secure digital (SD) card.

You can create as many AVDs as you want in order to test your applications with several different configurations. This testing is important to confirm the behavior of your application when it is run on different devices with varying capabilities.

NOTE *Appendix B will discuss some of the capabilities of the Android Emulator.*

To create an AVD, go to Windows ➪ Android SDK and AVD Manager.

Select the Available packages option in the left pane and expand the package name shown in the right pane. Figure 1-16 shows the various packages available for you to create AVDs to emulate the different versions of an Android device.



Android SDK and AVD Manager

Virtual devices
Installed packages
Available packages

SDK Location: C:\Users\Wei-Meng Lee\Desktop\Android 2.3\android-sdk-windows\

Packages available for download

```
▴ ✓ Android Repository
  › ✓ Android SDK Tools, revision 8
  › ✓ Android SDK Platform-tools, revision 1
  › ✓ Documentation for Android SDK, API 9, revision 1
  › ✓ SDK Platform Android 2.3, API 9, revision 1
  › ✓ SDK Platform Android 2.2, API 8, revision 2
  › ✓ SDK Platform Android 2.1, API 7, revision 2
  › ✓ SDK Platform Android 2.0.1, API 6, revision 1 (Obsolete)
  › ✓ SDK Platform Android 2.0, API 5, revision 1 (Obsolete)
  › ✓ SDK Platform Android 1.6, API 4, revision 3
  › ✓ SDK Platform Android 1.5, API 3, revision 4
  › ✓ SDK Platform Android 1.1, API 2, revision 1 (Obsolete)
  › ✓ Samples for SDK API 9, revision 1
  › ✓ Samples for SDK API 8, revision 1
  › ✓ Samples for SDK API 7, revision 1
▴ ✓ Third party Add-ons
  ▴ ✓ Google Inc. add-ons (dl-ssl.google.com)
    › ✓ Google APIs by Google Inc., Android API 9, revision 1
    › ✓ Google APIs by Google Inc., Android API 8, revision 2
    › ✓ Google APIs by Google Inc., Android API 7, revision 1
    › ✓ Google APIs by Google Inc., Android API 6, revision 1 (Obsolete)
    › ✓ Google APIs by Google Inc., Android API 5, revision 1 (Obsolete)
    › ✓ Google APIs by Google Inc., Android API 4, revision 2
    › ✓ Google APIs by Google Inc., Android API 3, revision 3
    › ✓ Google Usb Driver package, revision 4
    › ✓ Google Market Licensing package, revision 1
  ▴ ✓ Samsung Electronics add-ons (innovator.samsungmobile.com)
    › ✓ GALAXY Tab by Samsung Electronics., Android API 8, revision 1
```

Description

FIGURE 1-16

Check the relevant tools, documentation, and platforms you need for your project.

Once you have selected the items you want, click the Install Selected button to download them. Because it takes a while to download from Google's server, it is a good idea to download only whatever you need immediately, and download the rest when you have more time.

> ***NOTE*** *For a start, you should at least select the latest SDK platform. At the time of writing, the latest SDK platform is SDK Platform Android 2.3, API 9, revision 1.*

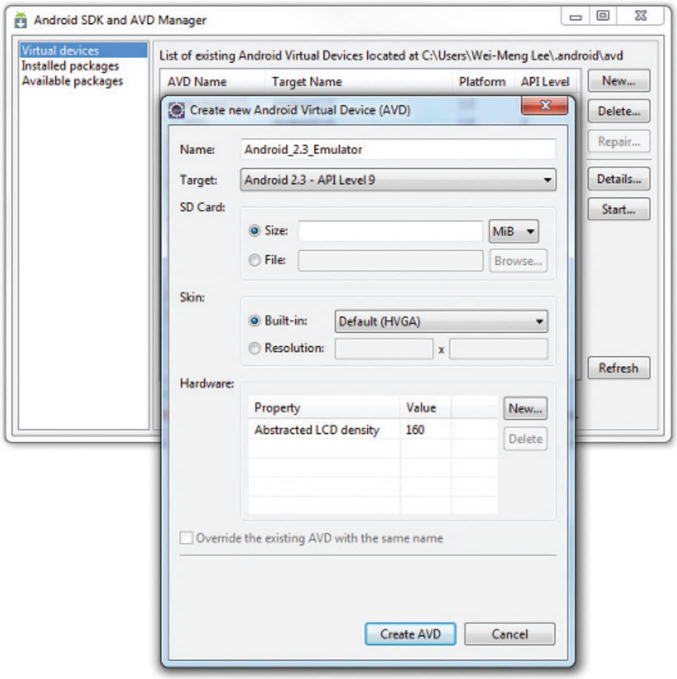
Each version of the Android OS is identified by an API level number. For example, Android 2.3 is level 9 (API 9), while Android 2.2 is level 8 (API 8), and so on. For each level, two platforms are available. For example, level 9 offers the following:

- SDK Platform Android 2.3
- Google APIs by Google Inc.

The key difference between the two is that the Google APIs platform contains the Google Maps library. Therefore, if the application you are writing requires Google Maps, you need to create an AVD using the Google APIs platform (more on this in Chapter 9, "Location Based Services."

Click the Virtual Devices item in the left pane of the window. Then click the New… button located in the right pane of the window.

In the Create new Android Virtual Device (AVD) window, enter the items as shown in Figure 1-17. Click the Create AVD button when you are done.



List of existing Android Virtual Devices located at C:\Users\Wei-Meng Lee\.android\avd

Name: Android_2.3_Emulator

Target: Android 2.3 - API Level 9

SD Card:
- (●) Size: MiB
- ( ) File:

Skin:
- (●) Built-in: Default (HVGA)
- ( ) Resolution: x

Hardware:

Property	Value
Abstracted LCD density	160

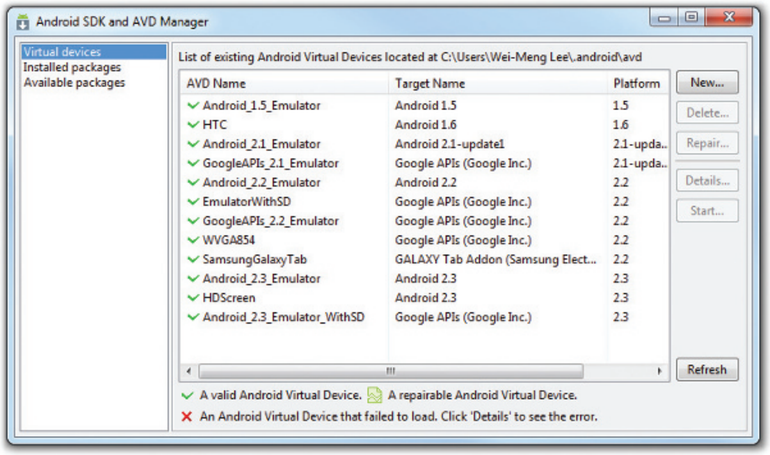
☐ Override the existing AVD with the same name

FIGURE 1-17

In this case, you have created an AVD (put simply, an Android emulator) that emulates an Android device running version 2.3 of the OS. In addition to what you have created, you also have the option to emulate the device with an SD card and different screen densities and resolutions.

NOTE *Appendix B explains how to emulate the different types of Android devices.*

It is preferable to create a few AVDs with different API levels so that your application can be tested on different devices. The example shown in Figure 1-18 shows the many AVDs created to test your applications on a wide variety of different Android platforms.



List of existing Android Virtual Devices located at C:\Users\Wei-Meng Lee\.android\avd

AVD Name	Target Name	Platform
✓ Android_1.5_Emulator	Android 1.5	1.5
✓ HTC	Android 1.6	1.6
✓ Android_2.1_Emulator	Android 2.1-update1	2.1-upda..
✓ GoogleAPIs_2.1_Emulator	Google APIs (Google Inc.)	2.1-upda..
✓ Android_2.2_Emulator	Android 2.2	2.2
✓ EmulatorWithSD	Google APIs (Google Inc.)	2.2
✓ GoogleAPIs_2.2_Emulator	Google APIs (Google Inc.)	2.2
✓ WVGA854	Google APIs (Google Inc.)	2.2
✓ SamsungGalaxyTab	GALAXY Tab Addon (Samsung Elect...	2.2
✓ Android_2.3_Emulator	Android 2.3	2.3
✓ HDScreen	Android 2.3	2.3
✓ Android_2.3_Emulator_WithSD	Google APIs (Google Inc.)	2.3

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

FIGURE 1-18

Creating Your First Android Application

With all the tools and the SDK downloaded and installed, it is now time to start your engine! As in all programming books, the first example uses the ubiquitous Hello World application. This will enable you to have a detailed look at the various components that make up an Android project.

So, without any further ado, let's dive straight in!

TRY IT OUT Creating Your First Android Application

codefile HelloWorld.zip available for download at Wrox.com

1. Using Eclipse, create a new project by selecting File ➪ Project… (see Figure 1-19).

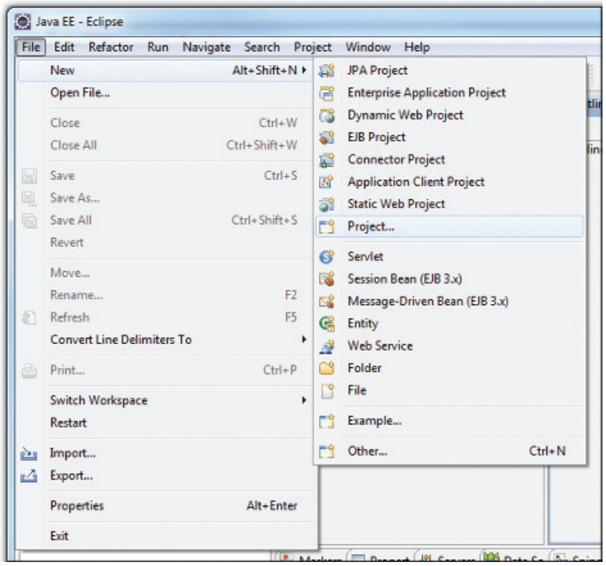


FIGURE 1-19

NOTE *After you have created your first Android application, subsequent Android projects can be created by selecting File ➪ New ➪ Android Project.*

2. Expand the Android folder and select Android Project (see Figure 1-20).

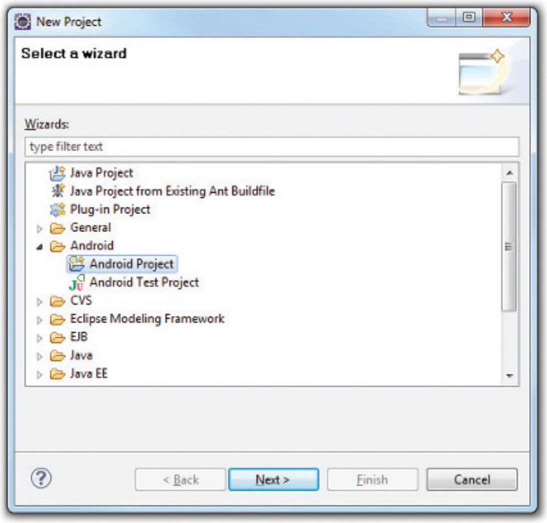
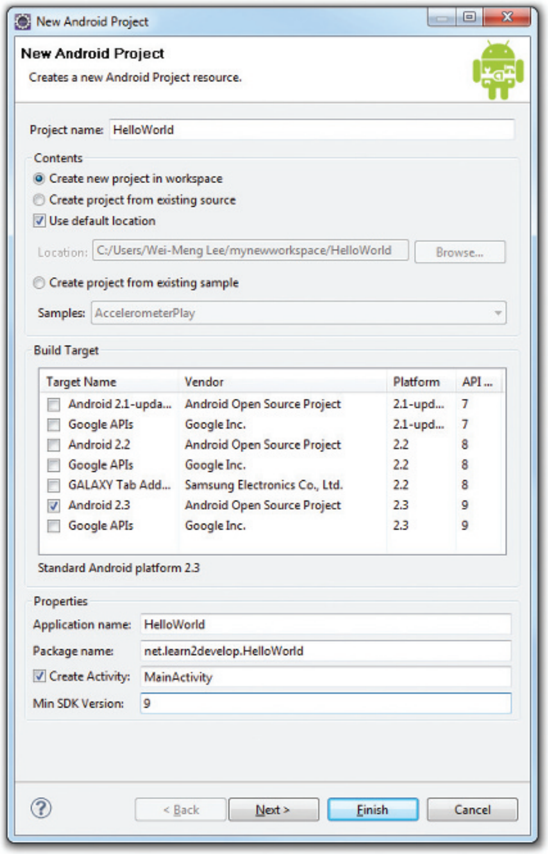


FIGURE 1-20

3. Name the Android project as shown in Figure 1-21 and then click Finish.


New Android Project

New Android Project

Creates a new Android Project resource.

Project name: HelloWorld

Contents

- ● Create new project in workspace
- ○ Create project from existing source
- ☑ Use default location

Location: C:/Users/Wei-Meng Lee/mynewworkspace/HelloWorld

- ○ Create project from existing sample

Samples: AccelerometerPlay

Build Target

	Target Name	Vendor	Platform	API ...
☐	Android 2.1-upda...	Android Open Source Project	2.1-upd...	7
☐	Google APIs	Google Inc.	2.1-upd...	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	GALAXY Tab Add...	Samsung Electronics Co., Ltd.	2.2	8
☑	Android 2.3	Android Open Source Project	2.3	9
☐	Google APIs	Google Inc.	2.3	9

Standard Android platform 2.3

Properties

Application name: HelloWorld

Package name: net.learn2develop.HelloWorld

☑ Create Activity: MainActivity

Min SDK Version: 9

FIGURE 1-21

NOTE *You need to have at least a period (.) in the package name. The recommended convention for the package name is to use your domain name in reverse order, followed by the project name. For example, my company's domain name is* `learn2develop.net`, *hence my package name would be* `net.learn2develop.HelloWorld`.

4. The Eclipse IDE should now look like Figure 1-22.
5. In the Package Explorer (located on the left of the Eclipse IDE), expand the HelloWorld project by clicking on the various arrows displayed to the left of each item in the project. In the `res/layout` folder, double-click the `main.xml` file (see Figure 1-23).

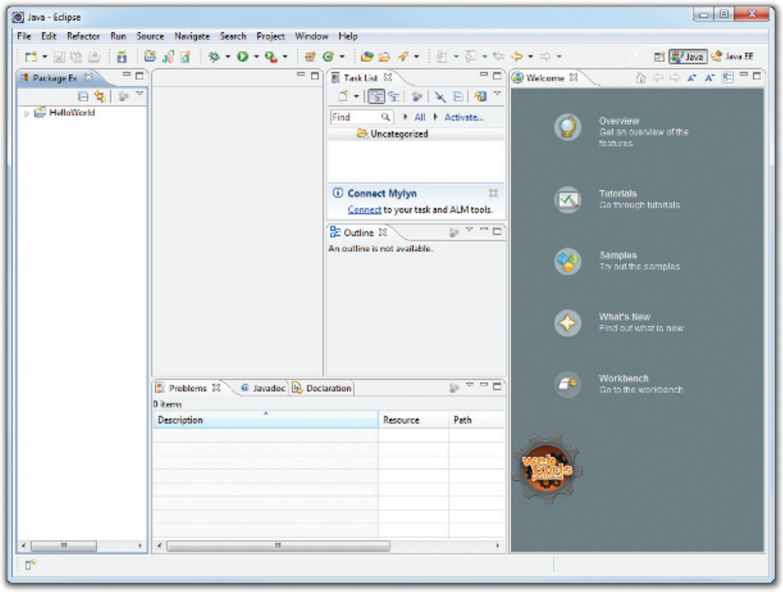


FIGURE 1-22

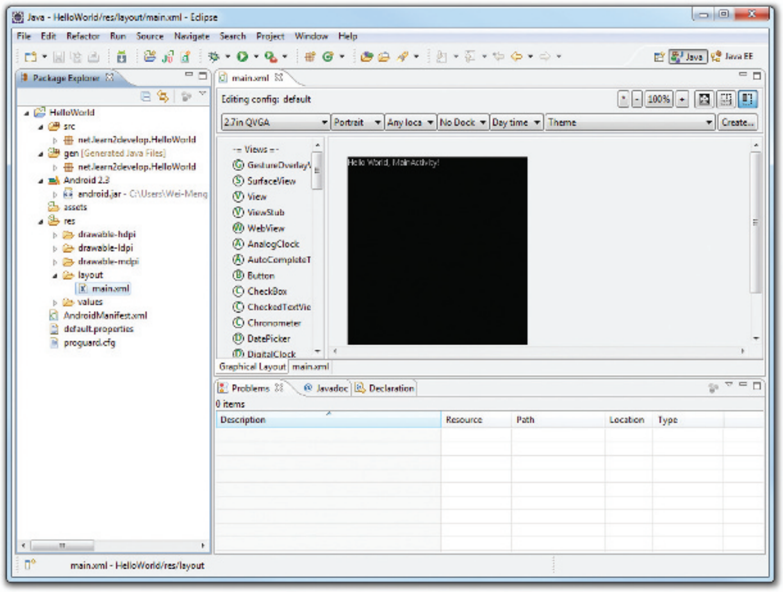
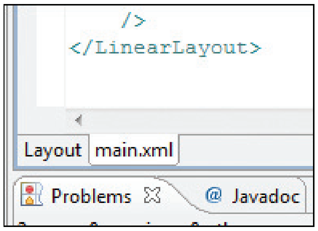


FIGURE 1-23

6. The `main.xml` file defines the user interface (UI) of your application. The default view is the Layout view, which lays out the activity graphically. To modify the UI, click the `main.xml` tab located at the bottom (see Figure 1-24).



```
    />
</LinearLayout>
```

FIGURE 1-24

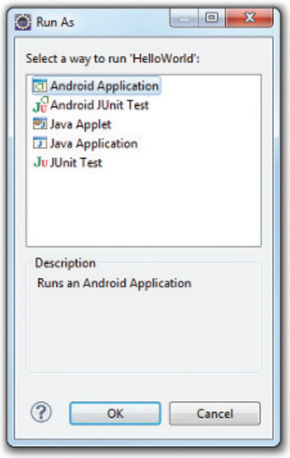
7. Add the following code in bold to the `main.xml` file:

```
<?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" >

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

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="This is my first Android Application!" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="And this is a clickable button!" />

</LinearLayout>
```

8. To save the changes made to your project, press Ctrl+s.

9. You are now ready to test your application on the Android Emulator. Select the project name in Eclipse and press F11. You will be asked to select a way to debug the application. Select Android Application as shown in Figure 1-25 and click OK.

NOTE *Some Eclipse installations have an irritating bug: After creating a new project, Eclipse reports that it contains errors when you try to debug the application. This happens even when you have not modified any files or folders in the project. To solve this problem, simply delete the* `R.java` *file located under the* `gen/net.learn2develop.HelloWorld` *folder; Eclipse will automatically generate a new* `R.java` *file for you. Once this is done, the project shouldn't contain any errors.*


Run As

Select a way to run 'HelloWorld':

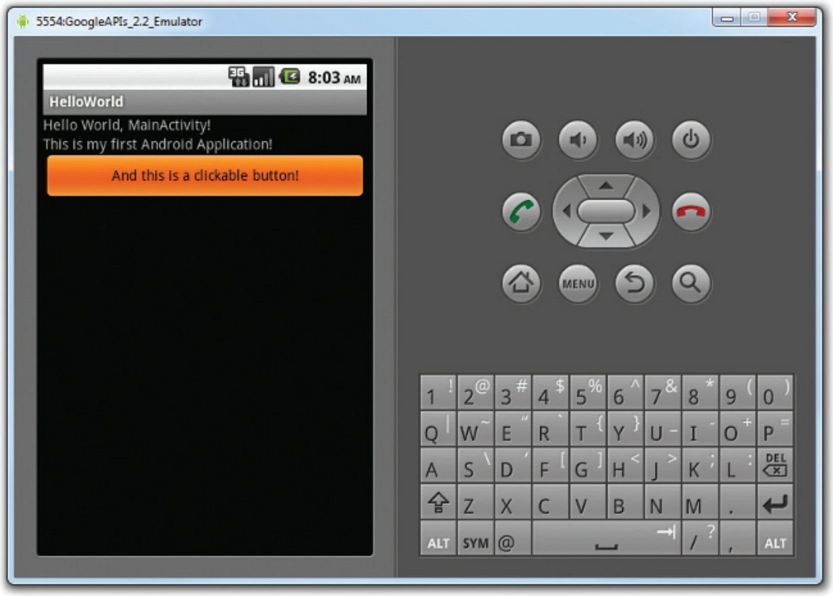
- Android Application
- Android JUnit Test
- Java Applet
- Java Application
- JUnit Test

Description

Runs an Android Application

FIGURE 1-25

10. The Android Emulator will now be started (if the emulator is locked, you need to slide the unlock button to unlock it first). Figure 1-26 shows the application running on the Android Emulator.


5554:GoogleAPIs_2.2_Emulator

8:03 AM

HelloWorld

Hello World, MainActivity!
This is my first Android Application!

FIGURE 1-26

11. Click the Home button (the house icon in the lower-left corner above the keyboard) so that it now shows the Home screen (see Figure 1-27).

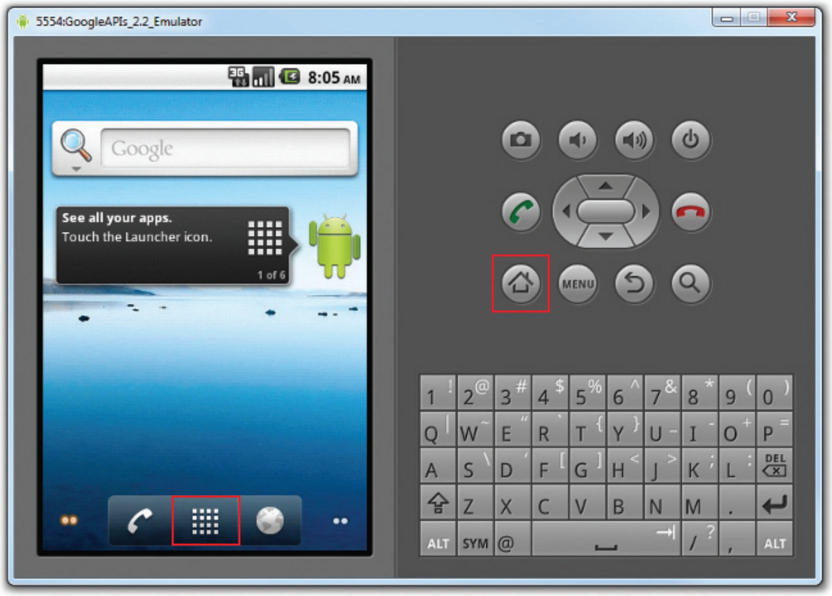


FIGURE 1-27

12. Click the application Launcher icon to display the list of applications installed on the device. Note that the HelloWorld application is now installed in the application launcher (see Figure 1-28).

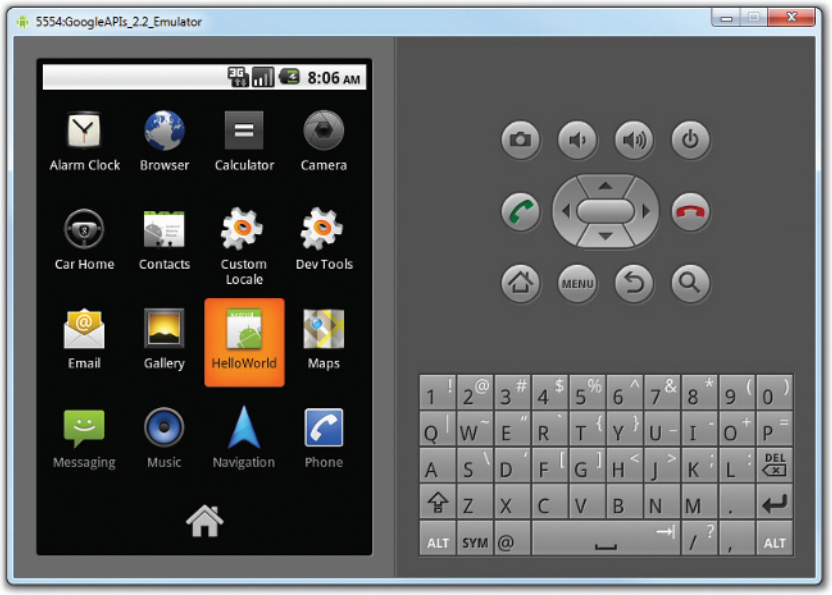
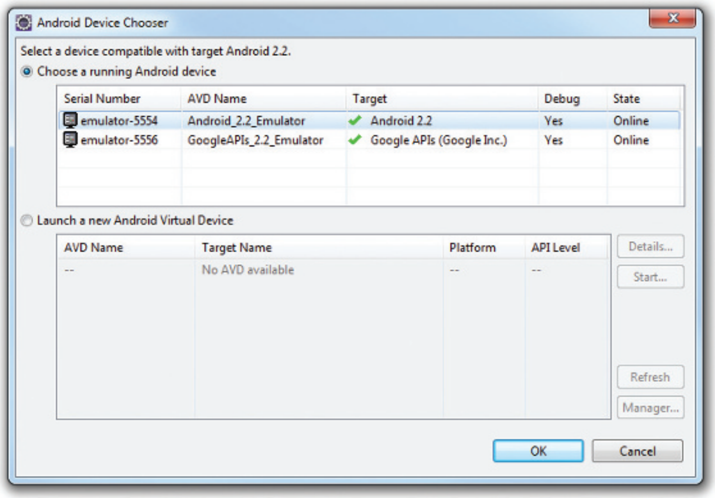


FIGURE 1-28

WHICH AVD WILL BE USED TO TEST YOUR APPLICATION?

Recall that earlier you created a few AVDs using the AVD Manager. So which one will be launched by Eclipse when you run an Android application? Eclipse will check the target that you specified (when you created a new project), comparing it against the list of AVDs that you have created. The first one that matches will be launched to run your application.

If you have more than one suitable AVD running prior to debugging the application, Eclipse will display the Android Device Chooser window, which enables you to select the desired emulator/device to debug the application (see Figure 1-29).



Select a device compatible with target Android 2.2.

[✓ Choose a running Android device]

Serial Number	AVD Name	Target	Debug	State
emulator-5554	Android_2.2_Emulator	✓ Android 2.2	Yes	Online
emulator-5556	GoogleAPIs_2.2_Emulator	✓ Google APIs (Google Inc.)	Yes	Online

Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level
--	No AVD available	--	--

FIGURE 1-29

How It Works

To create an Android project using Eclipse, you need to supply the information shown in Table 1-2.

TABLE 1-2: Project Files Created by Default

PROPERTIES	DESCRIPTION
Project name	The name of the project
Application name	A user-friendly name for your application
Package name	The name of the package. You should use a reverse domain name for this.
Create Activity	The name of the first activity in your application
Min SDK Version	The minimum version of the SDK that your project is targeting

In Android, an Activity is a window that contains the user interface of your applications. An application can have zero or more activities; in this example, the application contains one activity: `MainActivity`. This `MainActivity` is the entry point of the application, which is displayed when the application is started. Chapter 2 discusses activities in more detail.

In this simple example, you modified the `main.xml` file to display the string "This is my first Android Application!" and a button. The `main.xml` file contains the user interface of the activity, which is displayed when `MainActivity` is loaded.

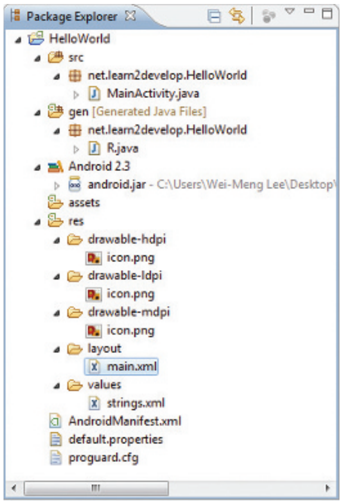
When you debug the application on the Android Emulator, the application is automatically installed on the emulator. And that's it — you have developed your first Android application!

The next section unravels how all the various files in your Android project work together to make your application come alive.

Anatomy of an Android Application

Now that you have created your first Hello World Android application, it is time to dissect the innards of the Android project and examine all the parts that make everything work.

First, note the various files that make up an Android project in the Package Explorer in Eclipse (see Figure 1-30).



```
Package Explorer
HelloWorld
  src
    net.learn2develop.HelloWorld
      MainActivity.java
  gen [Generated Java Files]
    net.learn2develop.HelloWorld
      R.java
  Android 2.3
    android.jar - C:\Users\Wei-Meng Lee\Desktop\
  assets
  res
    drawable-hdpi
      icon.png
    drawable-ldpi
      icon.png
    drawable-mdpi
      icon.png
    layout
      main.xml
    values
      strings.xml
  AndroidManifest.xml
  default.properties
  proguard.cfg
```

FIGURE 1-30

The various folders and their files are as follows:

- `src` — Contains the `.java` source files for your project. In this example, there is one file, `MainActivity.java`. The `MainActivity.java` file is the source file for your activity. You will write the code for your application in this file.
- `Android 2.3` library — This item contains one file, `android.jar`, which contains all the class libraries needed for an Android application.
- `gen` — Contains the `R.java` file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
- `assets` — This folder contains all the assets used by your application, such as HTML, text files, databases, etc.
- `res` — This folder contains all the resources used in your application. It also contains a few other subfolders: `drawable-<`*`resolution`*`>`, `layout`, and `values`. Chapter 3 talks more about how you can support devices with different screen resolutions and densities.
- `AndroidManifest.xml` — This is the manifest file for your Android application. Here you specify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.). Chapter 2 discusses the use of the `AndroidManifest.xml` file in more details.

The `main.xml` file defines the user interface for your activity. Observe the following in bold:

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

The `@string` in this case refers to the `strings.xml` file located in the `res/values` folder. Hence, `@string/hello` refers to the `hello` string defined in the `strings.xml` file, which is "Hello World, MainActivity!":

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">HelloWorld</string>
</resources>
```

It is recommended that you store all the string constants in your application in this `strings.xml` file and reference these strings using the `@string` identifier. That way, if you ever need to localize your application to another language, all you need to do is replace the strings stored in the `strings.xml` file with the targeted language and recompile your application.

Observe the content of the `AndroidManifest.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.HelloWorld"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

The `AndroidManifest.xml` file contains detailed information about the application:

- It defines the package name of the application as `net.learn2develop.HelloWorld`.
- The version code of the application is 1. This value is used to identify the version number of your application. It can be used to programmatically determine whether an application needs to be upgraded.
- The version name of the application is 1.0. This string value is mainly used for display to the user. You should use the format: *<major>.<minor>.<point>* for this value.
- The application uses the image named `icon.png` located in the `drawable` folder.

- The name of this application is the string named `app_name` defined in the `strings.xml` file.
- There is one activity in the application represented by the `MainActivity.java` file. The label displayed for this activity is the same as the application name.
- Within the definition for this activity, there is an element named `<intent-filter>`:
    - The action for the intent filter is named `android.intent.action.MAIN` to indicate that this activity serves as the entry point for the application.
    - The category for the intent-filter is named `android.intent.category.LAUNCHER` to indicate that the application can be launched from the device's Launcher icon. Chapter 2 discusses intents in more details.
- Finally, the `android:minSdkVersion` attribute of the `<uses-sdk>` element specifies the minimum version of the OS on which the application will run.

As you add more files and folders to your project, Eclipse will automatically generate the content of `R.java`, which at the moment contains the following:

```
package net.learn2develop.HelloWorld;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

You are not supposed to modify the content of the `R.java` file; Eclipse automatically generates the content for you when you modify your project.

NOTE *If you delete* `R.java` *manually, Eclipse will regenerate it for you immediately. Note that in order for Eclipse to generate the* `R.java` *file for you, the project must not contain any errors. If you realize that Eclipse has not regenerated* `R.java` *after you have deleted it, check your project again. The code may contain syntax errors, or your XML files (such as* `AndroidManifest.xml`, `main.xml`, *etc.) may not be well-formed.*

Finally, the code that connects the activity to the UI (main.xml) is the setContentView() method, which is in the MainActivity.java file:

```
package net.learn2develop.HelloWorld;

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

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Here, R.layout.main refers to the main.xml file located in the res/layout folder. As you add additional XML files to the res/layout folder, the filenames will automatically be generated in the R.java file. The onCreate() method is one of many methods that are fired when an activity is loaded. Chapter 2 discusses the life cycle of an activity in more detail.

SUMMARY

This chapter has provided a brief overview of Android, and highlighted some of its capabilities. If you have followed the sections on downloading the tools and SDK, you should now have a working system — one that is capable of developing more interesting Android applications other than the Hello World application. In the next chapter, you will learn about the concepts of activities and intents, and the very important roles they play in Android.

EXERCISES

1. What is an AVD?
2. What is the difference between the android:versionCode and android:versionName attributes in the AndroidManifest.xml file?
3. What is the use of the strings.xml file?

Answers to the Exercises can be found in Appendix C.

▶ WHAT YOU LEARNED IN THIS CHAPTER

TOPIC	KEY CONCEPTS
Android OS	Android is an open-source mobile operating system based on the Linux operating system. It is available to anyone who wants to adapt it to run on their own devices.
Languages used for Android application development	You use the Java programming language to develop Android applications. Written applications are compiled into Dalvik executables, which are then run on top of the Dalvik Virtual Machine.
Android Market	The Android Market hosts all the various Android applications written by third-party developers.
Tools for Android Application Development	Eclipse IDE, Android SDK, and the ADT
Activity	An activity is represented by a screen in your Android application. Each application can have zero or more activities.
The Android manifest file	The `AndroidManifest.xml` file contains detailed configuration information for your application. As your application gets more sophisticated, you will modify this file, and you will see the different information you can add to this file as you progress through the chapters.