Free and Open Source

Android is an open source platform. Neither developers nor handset manufacturers pay royalties or license fees to develop for the platform.

The underlying operating system of Android is licensed under GNU General Public License Version 2 (GPLv2), a strong "copyleft" license where any third-party improvements must continue to fall under the open source licensing agreement terms. The Android framework is distributed under the Apache Software License (ASL/Apache2), which allows for the distribution of both open- and closed-source derivations of the source code. Commercial developers (handset manufacturers especially) can choose to enhance the platform without having to provide their improvements to the open source community. Instead, developers can profit from enhancements such as handset-specific improvements and redistribute their work under whatever licensing they want.

Android application developers have the ability to distribute their applications under whatever licensing scheme they prefer. Developers can write open source freeware or traditional licensed applications for profit and everything in between.

Familiar and Inexpensive Development Tools

Unlike some proprietary platforms that require developer registration fees, vetting, and expensive compilers, there are no upfront costs to developing Android applications.

Freely Available Software Development Kit

The Android SDK and tools are freely available. Developers can download the Android SDK from the Android website after agreeing to the terms of the Android Software Development Kit License Agreement.

Familiar Language, Familiar Development Environments

Developers have several choices when it comes to integrated development environments (IDEs). Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications. Eclipse is the most popular IDE for Android development, and there is an Android plug-in available for facilitating Android development. Android applications can be developed on the following operating systems:

- Windows XP (32-bit) or Vista (32-bit or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Linux Ubuntu 8.04 LTS, Hardy Heron)

Reasonable Learning Curve for Developers

Android applications are written in a well-respected programming language: Java.

The Android application framework includes traditional programming constructs, such as threads and processes and specially designed data structures to encapsulate objects commonly used in mobile applications. Developers can rely on familiar class libraries, such as `java.net` and `java.text`. Specialty libraries for tasks such as graphics and database

management are implemented using well-defined open standards such as OpenGL Embedded Systems (OpenGL ES) or SQLite.

Enabling Development of Powerful Applications

In the past, handset manufacturers often established special relationships with trusted third-party software developers (OEM/ODM relationships). This elite group of software developers wrote native applications, such as messaging and web browsers, which shipped on the handset as part of the phone's core feature set. To design these applications, the manufacturer would grant the developer privileged inside access and knowledge of a handset's internal software framework and firmware.

On the Android platform, there is no distinction between native and third-party applications, enabling healthy competition among application developers. All Android applications use the same libraries. Android applications have unprecedented access to the underlying hardware, allowing developers to write much more powerful applications. Applications can be extended or replaced altogether. For example, Android developers are now free to design email clients tailored to specific email servers, such as Microsoft Exchange or Lotus Notes.

Rich, Secure Application Integration

Recall from the bat story I previously shared that I accessed a variety of phone applications in the course of a few moments: text messaging, phone dialer, camera, email, picture messaging, and the browser. Each was a separate application running on the phone—some built-in and some purchased. Each had its own unique user interface. None were truly integrated.

Not so with Android. One of the Android platform's most compelling and innovative features is well-designed application integration. Android provides all the tools necessary to build a better "bat trap," if you will, by allowing developers to write applications that seamlessly leverage core functionality such as web browsing, mapping, contact management, and messaging. Applications can also become content providers and share their data among each other in a secure fashion.

Platforms such as Symbian have suffered from setbacks due to malware. Android's vigorous application security model helps protect the user and the system from malicious software.

No Costly Obstacles to Publication

Android applications have none of the costly and time-intensive testing and certification programs required by other platforms such as BREW and Symbian.

A "Free Market" for Applications

Android developers are free to choose any kind of revenue model they want. They can develop freeware, shareware, or trial-ware applications, ad-driven, and paid applications. Android was designed to fundamentally change the rules about what kind of wireless applications could be developed. In the past, developers faced many restrictions that had little to do with the application functionality or features:

- Store limitations on the number of competing applications of a given type
- Store limitations on pricing, revenue models, and royalties
- Operator unwillingness to provide applications for smaller demographics

With Android, developers can write and successfully publish any kind of application they want. Developers can tailor applications to small demographics, instead of just large-scale money-making ones often insisted upon by mobile operators. Vertical market applications can be deployed to specific, targeted users.

Because developers have a variety of application distribution mechanisms to choose from, they can pick the methods that work for them instead of being forced to play by others' rules. Android developers can distribute their applications to users in a variety of ways:

- Google developed the Android Market (see Figure 1.7), a generic Android application store with a revenue-sharing model.

Figure 1.7 The Android market.

- Handango.com added Android applications to its existing catalogue using their billing models and revenue-sharing model.
- Developers can come up with their own delivery and payment mechanisms.

Mobile operators are still free to develop their own application stores and enforce their own rules, but it will no longer be the only opportunity developers have to distribute their applications.

A New and Growing Platform

Android might be the next generation in mobile platforms, but the technology is still in its early stages. Early Android developers have had to deal with the typical roadblocks associated with a new platform: frequently revised SDKs, lack of good documentation, and market uncertainties.

On the other hand, developers diving into Android development now benefit from the first-to-market competitive advantages we've seen on other platforms such as BREW

and Symbian. Early developers who give feedback are more likely to have an impact on the long-term design of the Android platform and what features will come in the next version of the SDK. Finally, the Android forum community is lively and friendly. Incentive programs, such as the ADC, have encouraged many new developers to dig into the platform.

Each new version of the Android SDK has provided a number of substantial improvements to the platform. In recent revisions, the Android platform has received some much-needed UI "polish," both in terms of visual appeal and performance. Although most of these upgrades and improvements were welcome and necessary, new SDK versions often cause some upheaval within the Android developer community. A number of published applications have required retesting and resubmission to the Android Marketplace to conform to new SDK requirements, which are quickly rolled out to all Android phones in the field as a firmware upgrade, rendering older applications obsolete.

Some older Android handsets are not capable of running the latest versions of the platform. This means that Android developers often need to target several different SDK versions to reach all users. Luckily, the Android development tools make this easier than ever.

The Android Platform

Android is an operating system and a software platform upon which applications are developed. A core set of applications for everyday tasks, such as web browsing and email, are included on Android handsets.

As a product of the OHA's vision for a robust and open source development environment for wireless, Android is an emerging mobile development platform. The platform was designed for the sole purpose of encouraging a free and open market that all mobile applications phone users might want to have and software developers might want to develop.

Android's Underlying Architecture

The Android platform is designed to be more fault-tolerant than many of its predecessors. The handset runs a Linux operating system upon which Android applications are executed in a secure fashion. Each Android application runs in its own virtual machine (see Figure 1.8). Android applications are managed code; therefore, they are much less likely to cause the phone to crash, leading to fewer instances of device corruption (also called "bricking" the phone, or rendering it useless).

The Linux Operating System

The Linux 2.6 kernel handles core system services and acts as a hardware abstraction layer (HAL) between the physical hardware of the handset and the Android software stack.

Some of the core functions the kernel handles include

- Enforcement of application permissions and security
- Low-level memory management

- Process management and threading
- The network stack
- Display, keypad input, camera, Wi-Fi, Flash memory, audio, and binder (IPC) driver access

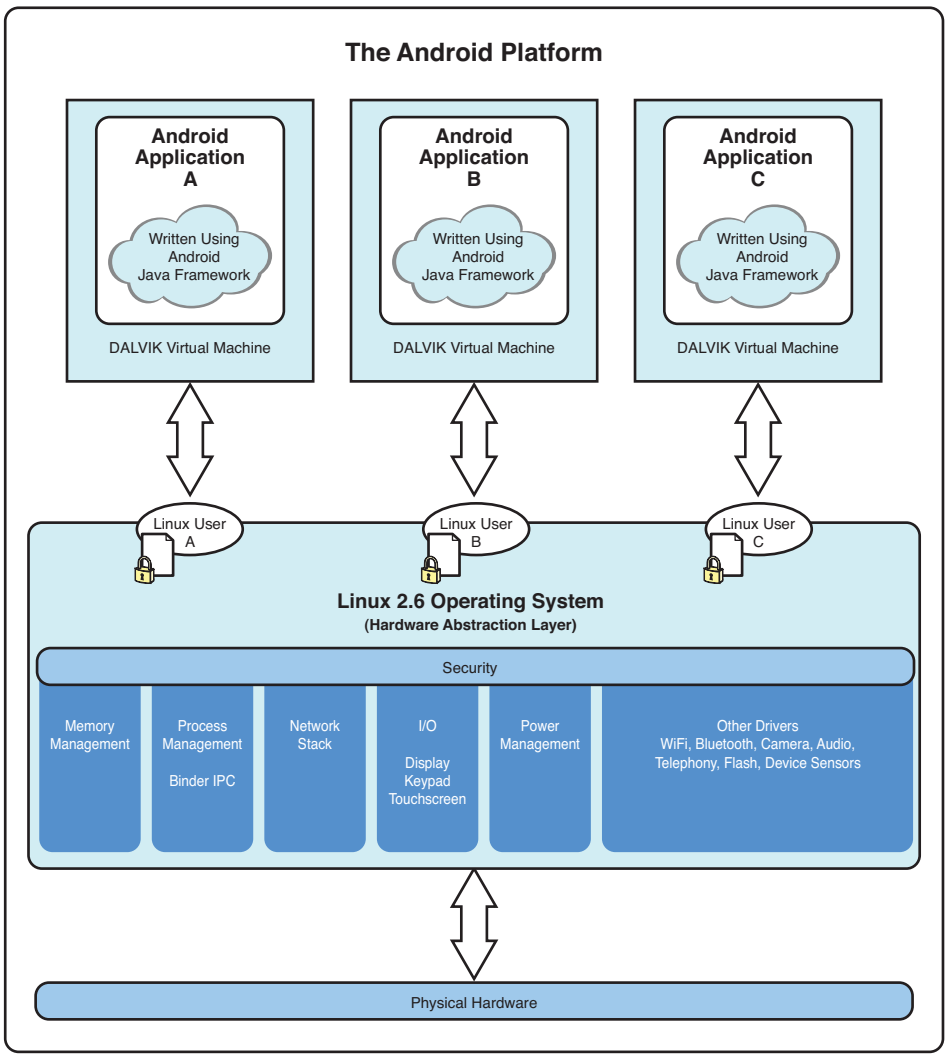

Figure 1.8 Diagram of the Android platform architecture.

Android Application Runtime Environment

Each Android application runs in a separate process, with its own instance of the Dalvik virtual machine (VM). Based on the Java VM, the Dalvik design has been optimized for mobile devices. The Dalvik VM has a small memory footprint, and multiple instances of the Dalvik VM can run concurrently on the handset.

Security and Permissions

The integrity of the Android platform is maintained through a variety of security measures. These measures help ensure that the user's data is secure and that the device is not subjected to malware.

Applications as Operating System Users

When an application is installed, the operating system creates a new user profile associated with the application. Each application runs as a different user, with its own private files on the file system, a user ID, and a secure operating environment.

The application executes in its own process with its own instance of the Dalvik VM and under its own user ID on the operating system.

Explicitly Defined Application Permissions

To access shared resources on the system, Android applications register for the specific privileges they require. Some of these privileges enable the application to use phone functionality to make calls, access the network, and control the camera and other hardware sensors. Applications also require permission to access shared data containing private and personal information, such as user preferences, user's location, and contact information.

Applications might also enforce their own permissions by declaring them for other applications to use. The application can declare any number of different permission types, such as read-only or read-write permissions, for finer control over the application.

Limited Ad-Hoc Permissions

Applications that act as content providers might want to provide some on-the-fly permissions to other applications for specific information they want to share openly. This is done using ad-hoc granting and revoking of access to specific resources using Uniform Resource Identifiers (URIs).

URIs index specific data assets on the system, such as images and text. Here is an example of a URI that provides the phone numbers of all contacts:

```
content://contacts/phones
```

To understand how this permission process works, let's look at an example.

Let's say we have an application that keeps track of the user's public and private birthday wish lists. If this application wanted to share its data with other applications, it could grant URI permissions for the public wish list, allowing another application permission to access this list without explicitly having to ask for it.

Application Signing for Trust Relationships

All Android applications packages are signed with a certificate, so users know that the application is authentic. The private key for the certificate is held by the developer. This helps establish a trust relationship between the developer and the user. It also enables the developer to control which applications can grant access to one another on the system. No certificate authority is necessary; self-signed certificates are acceptable.

Marketplace Developer Registration

To publish applications on the popular Android Market, developers must create a developer account. The Android Market is managed closely and no malware is tolerated.

Developing Android Applications

The Android SDK provides an extensive set of application programming interfaces (APIs) that is both modern and robust. Android handset core system services are exposed and accessible to all applications. When granted the appropriate permissions, Android applications can share data among one another and access shared resources on the system securely.

Android Programming Language Choices

Android applications are written in Java (see Figure 1.9). For now, the Java language is the developer's only choice on the Android platform.

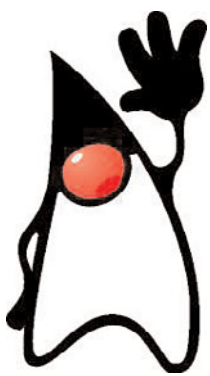

Figure 1.9 Duke, the Java mascot.

There has been some speculation that other programming languages, such as C++, might be added in future versions of Android. If your application must rely on native code in another language such as C or C++, you might want to consider integrating it using the Android Native Development Kit (NDK). We talk more about this in Chapter 18, "Using the Android NDK."

No Distinctions Made Between Native and Third-Party Applications

Unlike other mobile development platforms, there is no distinction between native applications and developer-created applications on the Android platform. Provided the application is granted the appropriate permissions, all applications have the same access to core libraries and the underlying hardware interfaces.

Android handsets ship with a set of native applications such as a web browser and contact manager. Third-party applications might integrate with these core applications, extend them to provide a rich user experience, or replace them entirely with alternative applications.

Commonly Used Packages

With Android, mobile developers no longer have to reinvent the wheel. Instead, developers use familiar class libraries exposed through Android's Java packages to perform common tasks such as graphics, database access, network access, secure communications, and utilities (such as XML parsing).

The Android packages include support for

- Common user interface widgets (Buttons, Spin Controls, Text Input)
- User interface layout
- Secure networking and web browsing features (SSL, WebKit)
- Structured storage and relational databases (SQLite)
- Powerful 2D and 3D graphics (including SGL and OpenGL ES)
- Audio and visual media formats (MPEG4, MP3, Still Images)
- Access to optional hardware such as location-based services (LBS), Wi-Fi, Bluetooth, and hardware sensors

Android Application Framework

The Android application framework provides everything necessary to implement your average application. The Android application lifecycle involves the following key components:

- Activities are functions the application performs.
- Groups of views define the application's layout.
- Intents inform the system about an application's plans.
- Services allow for background processing without user interaction.
- Notifications alert the user when something interesting happens.

Android applications can interact with the operating system and underlying hardware using a collection of managers. Each manager is responsible for keeping the state of some underlying system service. For example, there is a `LocationManager` that facilitates interaction with the location-based services available on the handset. The `ViewManager` and `WindowManager` manage user interface fundamentals.

Applications can interact with one another by using or acting as a `ContentProvider`. Built-in applications such as the Contact manager are content providers, allowing third-party applications to access contact data and use it in an infinite number of ways. The sky is the limit.

Summary

Mobile software development has evolved over time. Android has emerged as a new mobile development platform, building on past successes and avoiding past failures of other platforms. Android was designed to empower the developer to write innovative applications. The platform is open source, with no up-front fees, and developers enjoy many benefits over other competing platforms. Now it's time to dive deeper and start writing Android code, so you can evaluate what Android can do for you.

References and More Information

Android Development:
http://developer.android.com
Open Handset Alliance:
http://www.openhandsetalliance.com

2

Setting Up Your Android Development Environment

Android developers write and test applications on their computers and then deploy those applications onto the actual device hardware for further testing.

In this chapter, you become familiar with all the tools you need master in order to develop Android applications. You also explore the Android Software Development Kit (SDK) installation and all it has to offer.

Configuring Your Development Environment

To write Android applications, you must configure your programming environment for Java development. The software is available online for download at no cost. Android applications can be developed on Windows, Macintosh, or Linux systems.

To develop Android applications, you need to have the following software installed on your computer:

- The **Java Development Kit** (JDK) Version 5 or 6, available for download at http://java.sun.com/javase/downloads/index.jsp.
- A compatible **Java IDE such as Eclipse** along with its JDT plug-in, available for download at http://www.eclipse.org/downloads/.
- The **Android SDK**, tools and documentation, available for download at http://developer.android.com/sdk/index.html.
- The **Android Development Tools** (ADT) plug-in for Eclipse, available for download through the Eclipse software update mechanism. For instructions on how to install this plug-in, see http://developer.android.com/sdk/eclipse-adt.html. Although this tool is optional for development, we highly recommend it and will use its features frequently throughout this book.

A complete list of Android development system requirements is available at http://developer.android.com/sdk/requirements.html. Installation instructions are at http://developer.android.com/sdk/installing.html.

Tip

Most developers use the popular Eclipse Integrated Development Environment (IDE) for Android development. The Android development team has integrated the Android development tools directly into the Eclipse IDE. However, developers are not constrained to using Eclipse; they can also use other IDEs. For information on using other development environments, begin by reading http://developer.android.com/guide/developing/other-ide.html.

Configuring Your Operating System for Device Debugging

To install and debug Android applications on Android devices, you need to configure your operating system to access the phone via the USB cable (see Figure 2.1). On some operating systems, such as Mac OS, this may just work. However, for Windows installations, you need to install the appropriate USB driver. You can download the Windows USB driver from the following website: http://developer.android.com/sdk/win-usb.html.

Figure 2.1 Android application debugging using the emulator and an Android handset.

Configuring Your Android Hardware for Debugging

Android devices have debugging disabled by default. Your Android device must be enabled for debugging via a USB connection in order to develop applications and run them on the device.

First, you need to enable your device to install Android applications other than those from the Android Market. This setting is reached by selecting Home, Menu, Settings, Applications. Here you should check (enable) the option called Unknown Sources.

More important development settings are available on the Android device by selecting Home, Menu, Settings, Applications, Development (see Figure 2.2). Here you should enable the following options:

- **USB Debugging:** This setting enables you to debug your applications via the USB connection.

- **Stay Awake:** This convenient setting keeps the phone from sleeping in the middle of your development work, as long as the device is plugged in.
- **Allow Mock Locations:** This setting enables you to send mock location information to the phone for development purposes and is very convenient for applications using location-based services (LBS).

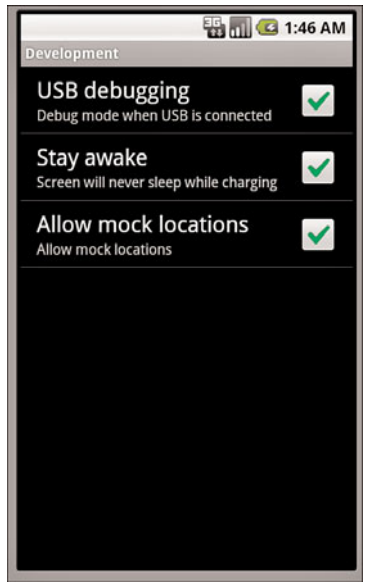

Figure 2.2 Android debug settings.

Upgrading the Android SDK

The Android SDK is upgraded from time to time. You can easily upgrade the Android SDK and tools from within Eclipse using the Android SDK and AVD Manager, which is installed as part of the ADT plug-in for Eclipse.

Changes to the Android SDK might include addition, update, and removal of features; package name changes; and updated tools. With each new version of the SDK, Google provides the following useful documents:

- **An Overview of Changes:** A brief description of major changes to the SDK.
- **An API Diff Report:** A complete list of specific changes to the SDK.
- **Release Notes:** A list of known issues with the SDK.

You can find out more about adding and updating SDK components at http://developer.android.com/sdk/adding-components.html.

Problems with the Android Software Development Kit

Because the Android SDK is constantly under active development, you might come across problems with the SDK. If you think you've found a problem, you can find a list of open issues and their status at the Android project Issue Tracker website. You can also submit new issues for review.

The Issue Tracker website for the Android open source project is http://code.google.com/p/android/issues/list. For more information about logging your own bugs or defects to be considered by the Android platform development team, check out the following website: http://source.android.com/source/report-bugs.html.

Tip

Frustrated with how long it takes for your bug to get fixed? It can be helpful to understand how the Android bug resolution process works. For more information on this process, see the following website: http://source.android.com/source/life-of-a-bug.html.

Exploring the Android SDK

The Android SDK comes with five major components: the Android SDK License Agreement, the Android Documentation, Application Framework, Tools, and Sample Applications.

Understanding the Android SDK License Agreement

Before you can download the Android SDK, you must review and agree to the Android SDK License Agreement. This agreement is a contract between you (the developer) and Google (copyright holder of the Android SDK).

Even if someone at your company has agreed to the Licensing Agreement on your behalf, it is important for you, the developer, to be aware of a few important points:

1. **Rights granted:** Google (as the copyright holder of Android) grants you a limited, worldwide, royalty-free, non-assignable, and non-exclusive license to use the SDK solely to develop applications for the Android platform. Google (and third-party contributors) are granting you license, but they still hold all copyrights and intellectual property rights to the material. Using the Android SDK does not grant you permission to use any Google brands, logos, or trade names. You will not remove any of the copyright notices therein. Third-party applications that your applications interact with (other Android apps) are subject to separate terms and fall outside this agreement.

2. **SDK usage:** You may only develop Android applications. You may not make derivative works from the SDK or distribute the SDK on any device or distribute part of the SDK with other software.

3. **SDK changes and backward compatibility:** Google may change the Android SDK at any time, without notice, without regard to backward compatibility. Although Android API changes were a major issue with prerelease versions of the

SDK, recent releases have been reasonably stable. That said, each SDK update does tend to affect a small number of existing applications in the field, necessitating updates.

4. **Android application developer rights:** You retain all rights to any Android software you develop with the SDK, including intellectual property rights. You also retain all responsibility for your own work.
5. **Android application privacy requirements:** You agree that your applications will protect the privacy and legal rights of its users. If your application uses or accesses personal and private information about the user (usernames, passwords, and so on), then your application will provide an adequate privacy notice and keep that data stored securely. Note that privacy laws and regulations may vary by user location; you as a developer are solely responsible for managing this data appropriately.
6. **Android application malware requirements:** You are responsible for all applications you develop. You agree not to write disruptive applications or malware. You are solely responsible for all data transmitted through your application.
7. **Additional terms for specific Google APIs:** Use of the Android Maps API is subject to further Terms of Service (specifically use of the following packages: `com.google.android.maps` and `com.android.location.Geocoder`). You must agree to these additional terms before using those specific APIs and always include the Google Maps copyright notice provided. Use of Google Data APIs (Google Apps such as Gmail, Blogger, Google Calendar, Google Finance Portfolio Data, Picasa, YouTube, and so on) is limited to access that the user has explicitly granted permission to your application by accepted permissions provided by the developer during installation time.
8. **Develop at your own risk:** Any harm that comes about from developing with the Android SDK is your own fault and not Google's.

Reading the Android SDK Documentation

A local copy of the Android documentation is provided in the /docs subfolder on disk (as shown in Figure 2.3).

The documentation is now divided into seven main sections:

- The **Home** tab is your general starting point within the Android documentation. Here you find developer announcements and important links to the latest hot topics in Android development.
- The **SDK** tab provides information about the different Android SDK versions available, as well as information about the Android Native Development Kit (NDK). You find the Android SDK release notes here as well.

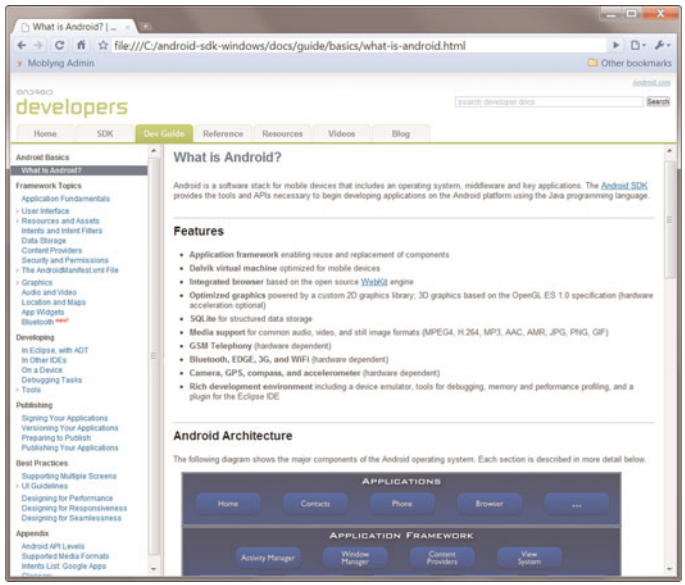

Figure 2.3 The Android SDK documentation.

- The **Dev Guide** tab introduces the Android platform and covers best practices for Android application design and development, as well as information about publishing applications.
- The **Reference** tab provides a drill-down listing of the Android APIs with detailed coverage of specific classes and interfaces.
- The **Resources** tab provides access to Android technical articles and tutorials. Here you also find links to the Android community online (groups, mailing list, and official Twitter feed), as well as the sample applications provided along with the Android SDK.
- The **Videos** tab provides access to online videos pertaining to Android development, including videos about the platform, developer tips, Android development sessions from the annual Google I/O conference, and developer sandbox interviews.
- The **Blog** tab provides access to the online blog published by the Android development team. Here you find announcements about SDK releases, helpful development tips, and notices of upcoming Android events.

The Android documentation is provided in HTML format locally and online at http://developer.android.com. Certain networked features of the Android documentation (such as the Blog and Video tabs) are only available online.

Exploring the Android Application Framework

The Android application framework is provided in the `android.jar` file. The Android SDK is made up of several important packages, as shown in Table 2.1.

Table 2.1 **Important Packages in the Android SDK**

Top-Level Package	Purpose
`android.*`	Android application fundamentals
`dalvik.*`	Dalvik Virtual Machine support classes
`java.*`	Core classes and familiar generic utilities for networking, security, math, and such
`javax.*`	Java extension classes including encryption support, parsers, SQL, and such
`junit.*`	Unit testing support
`org.apache.http.*`	Hypertext Transfer Protocol (HTTP) protocol
`org.json`	JavaScript Object Notation (JSON) support
`org.w3c.dom`	W3C Java bindings for the Document Object Model Core (XML and HTML)
`org.xml.sax.*`	Simple API for XML (SAX) support for XML
`org.xmlpull.*`	High-performance XML parsing

There is also an optional Google APIs Add-On, which is an extension to the Android SDK that helps facilitate development using Google Maps and other Google APIs and services. For example, if you want to include the MapView control in your application, you need to install and use this feature. This Add-On corresponds to the `com.google.*` package (including `com.google.android.maps`) and requires agreement to additional Terms of Service and registration for an API Key. For more information on the Google APIs Add-On, see http://code.google.com/android/add-ons/google-apis/.

Getting to Know the Android Tools

The Android SDK provides many tools to design, develop, debug, and deploy your Android applications. The Eclipse Plug-In incorporates many of these tools seamlessly into your development environment and provides various wizards for creating and debugging Android projects.

Settings for the ADT plug-in are found in Eclipse under Window, Preferences, Android. Here you can set the disk location where you installed the Android SDK and tools, as well as numerous other build and debugging settings.

The ADT plug-in adds a number of useful functions to the default Eclipse IDE. Several new buttons are available on the toolbar, including buttons to

- Launch the Android SDK and AVD Manager
- Create a new project using the Android Project Wizard
- Create a new test project using the Android Project Wizard
- Create a new Android XML resource file

These features are accessible through the Eclipse toolbar buttons shown in Figure 2.4.

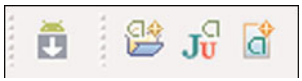

Figure 2.4 Android features added to the Eclipse toolbar.

There is also a special Eclipse perspective for debugging Android applications called DDMS (Dalvik Debug Monitor Server). You can switch to this perspective within Eclipse by choosing Window, Open Perspective, DDMS or by changing to the DDMS perspective in the top-right corner of the screen. We talk more about DDMS later in this chapter. After you have designed an Android application, you can also use the ADT plug-in for Eclipse to launch a wizard to package and sign your Android application for publication. We talk more about this in Chapter 29, "Selling Your Android Application."

Android SDK and AVD Manager

The Android SDK and AVD Manager, shown in Figure 2.5, is a tool integrated into Eclipse. This tool performs two major functions: management of multiple versions of the Android SDK on the development machine and management of the developer's Android Virtual Device (AVD) configurations.

Figure 2.5 The Android SDK and AVD Manager.

Much like desktop computers, different Android devices run different versions of the Android operating system. Developers need to be able to target different Android SDK

versions with their applications. Some applications target a specific Android SDK, whereas others try to provide simultaneous support for as many versions as possible.

The Android SDK and AVD Manager facilitate Android development across multiple platform versions simultaneously. When a new Android SDK is released, you can use this tool to download and update your tools while still maintaining backward compatibility and use older versions of the Android SDK.

The tool also manages the AVD configurations. To manage applications in the Android emulator, you must configure an AVD. This AVD profile describes what type of device you want the emulator to simulate, including which Android platform to support. You can specify different screen sizes and orientations, and you can specify whether the emulator has an SD card and, if so, what capacity.

Android Emulator

The Android emulator, shown in Figure 2.6, is one of the most important tools provided with the Android SDK. You will use this tool frequently when designing and developing Android applications. The emulator runs on your computer and behaves much as a mobile device would. You can load Android applications into the emulator, test, and debug them.

Figure 2.6 The Android emulator.

The emulator is a generic device and is not tied to any one specific phone configuration. You describe the hardware and software configuration details that the emulator is to simulate by providing an AVD configuration.

Tip

You should be aware that the Android emulator is a substitute for a real Android device, but it's an imperfect one. The emulator is a valuable tool for testing but cannot fully replace testing on actual target devices.

For more information about the emulator, see Appendix A, "The Android Emulator Quick-Start Guide." You can also find exhaustive information about the Android emulator in the Android SDK Documentation: http://developer.android.com/guide/developing/tools/emulator.html.

Dalvik Debug Monitor Server (DDMS)

The Dalvik Debug Monitor Server (DDMS) is a command-line tool that has also been integrated into Eclipse as a perspective (see Figure 2.7). This tool provides you with direct access to the device—whether it's the emulator virtual device or the physical device. You use DDMS to view and manage processes and threads running on the device, view heap data, attach to processes to debug, and a variety of other tasks.

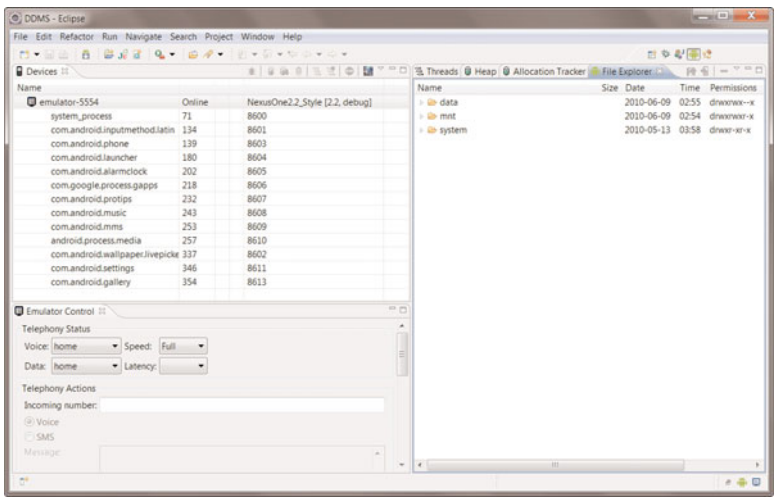

Figure 2.7 Using DDMS integrated into an Eclipse perspective.

For more information about the DDMS, see Appendix B, "The Android DDMS Quick-Start Guide." You can also find exhaustive details about DDMS in the Android SDK Documentation: http://developer.android.com/guide/developing/tools/ddms.html.

Android Debug Bridge (ADB)

The Android Debug Bridge (ADB) is a client-server tool used to enable developers to debug Android code on the emulator and the device using a standard Java IDE such as

Eclipse. The DDMS and the Android Development Plug-In for Eclipse both use the ADB to facilitate interaction between the development environment and the device (or emulator).

Developers can also use ADB to interact with the device file system, install Android applications manually, and issue shell commands. For example, the `sqlite3` shell commands enable you to access device database. The Application Exerciser Monkey commands generate random input and system events to stress test your application. One of the most important aspects of the ADB for the developer is its logging system (`Logcat`).

For more information about the ADB, see Appendix C, "The Android Debug Bridge Quick-Start Guide." For an exhaustive reference, see the Android SDK Documentation at http://developer.android.com/guide/developing/tools/adb.html.

Android Hierarchy Viewer

The Android Hierarchy Viewer (see Figure 2.8), a visual tool that illustrates layout component relationships, helps developers design and debug user interfaces. Developers can use this tool to inspect the `View` properties and develop pixel-perfect layouts. For more information about user interface design and the Hierarchy Viewer, see Chapter 8, "Designing User Interfaces with Layouts."

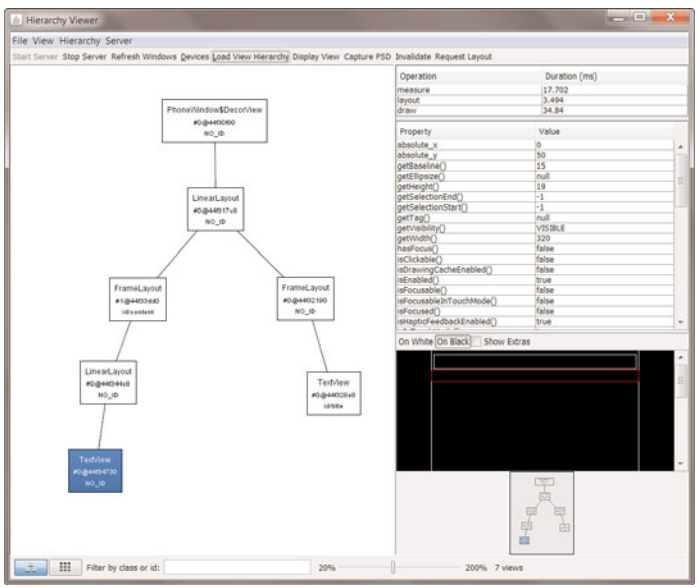

Figure 2.8 Screenshot of the Android Hierarchy Viewer in action.

Other Tools

Android SDK provides a number of other tools provided with the Android SDK. Many of these tools provide the underlying functionality that has been integrated into Eclipse using

the Android Development Tools (ADT) plug-in. However, if you are not using Eclipse, these tools may be used on the command-line.

Other tools are special-purpose utilities. For example, the Draw Nine-patch tool enables you to design stretchable PNG images, which is useful for supporting different screen sizes. Likewise, the layoutopt tool helps developers optimize their user interfaces for performance. We discuss a number of these special tools in later chapters as they become relevant.

You can read about all the Android tools in the SDK documentation at http://developer.android.com/guide/developing/tools/index.html.

Exploring the Android Sample Applications

The Android SDK provides many samples and demo applications to help you learn the ropes of Android Development. Many of these demo applications are provided as part of the Android SDK and are located in the /samples subdirectory of the Android SDK. You can find more sample applications on the Android Developer website under the Resources tab.

Tip

On some Android SDK installations, the sample applications must be downloaded separately by updating your SDK installation using the Android SDK and AVD Manager. All sample applications can also be found on the Android Developer website.

Some of the most straightforward demo applications to take a look at are

- **ApiDemos:** A menu-driven utility that demonstrates a wide variety of Android APIs, from user interface widgets to application lifecycle components such as services, alarms, and notifications. You can read a nice write-up about this application at http://developer.android.com/resources/samples/ApiDemos/.
- **Snake:** A simple game that demonstrates bitmap drawing and key events. You can find a nice write-up about this game at http://developer.android.com/resources/samples/Snake/.
- **NotePad:** A simple list application that demonstrates database access and Live Folder functionality. You can read a nice write-up about this application at http://developer.android.com/resources/samples/NotePad/.
- **LunarLander:** A simple game that demonstrates drawing and animation. You can find a nice write-up about this game at http://developer.android.com/resources/samples/LunarLander/.

There are numerous other sample applications, but they demonstrate very specific Android features that are discussed later in this book.

Summary

In this chapter, you installed, configured, and explored all the tools you need to start developing Android applications, including the appropriate JDK, the Eclipse development environment, and the Android SDK. You explored many of the tools provided along with the Android SDK and understand their functions. Finally, you perused the sample applications provided along with the Android SDK. You should now have a reasonable development environment configured to write Android applications.

References and More Information

Google's Android Developer's Guide:
http://developer.android.com/guide/index.html
Android SDK Download Site:
http://developer.android.com/sdk/
Android SDK License Agreement:
http://developer.android.com/sdk/terms.html
The Java Platform, Standard Edition:
http://java.sun.com/javase
The Eclipse Project:
http://www.eclipse.org