1

Hello, Android

Whether you're an experienced mobile engineer, a desktop or web developer, or a complete programming novice, Android represents an exciting new opportunity to write innovative applications for mobile devices.

Despite the name, Android will not help you create an unstoppable army of emotionless robot warriors on a relentless quest to cleanse the earth of the scourge of humanity. Instead, *Android* is an open source software stack that includes the operating system, middleware, and key applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets.

Small, stylish, and versatile, modern mobile phones have become powerful tools that incorporate cameras, media players, GPS systems, and touch screens. As technology has evolved, mobile devices have become about more than simply making calls, but their software and development platforms have struggled to keep pace.

Until recently, mobile phones were largely closed environments built on proprietary operating systems that required proprietary development tools. The phones themselves often prioritized native applications over those written by third parties. This has introduced an artificial barrier for developers hoping to build on increasingly powerful mobile hardware.

In Android, native and third-party applications are written using the same APIs and executed on the same run time. These APIs feature hardware access, location-based services, support for background services, map-based activities, relational databases, interdevice peer-to-peer messaging, and 2D and 3D graphics.

Using this book, you will learn how to use these APIs to create your own Android applications. In this chapter, you'll learn some mobile development guidelines and be introduced to the features available from the Android development platform.

Android has powerful APIs, excellent documentation, a thriving developer community, and no development or distribution costs. As mobile devices continue to increase in popularity, this is an exciting opportunity to create innovative mobile phone applications no matter what your development background.

A Little Background

In the days before Twitter and Facebook, when Google was still a twinkle in its founders' eyes and dinosaurs roamed the earth, mobile phones were just that — portable phones small enough to fit inside a briefcase, featuring batteries that could last up to several hours; they offered the freedom to make calls without being physically connected to a landline.

Increasingly small, stylish, and powerful mobile phones are now as ubiquitous as they are indispensable. Hardware advancements have made mobiles smaller and more efficient while including an increasing number of peripherals.

Beginning with cameras and media players, mobiles now include GPS systems, accelerometers, and touch screens. While these hardware innovations should prove fertile ground for software development, the applications available for mobile phones have generally lagged behind the hardware.

The Not So Distant Past

Historically, developers, generally coding in low-level C or C++, have needed to understand the specific hardware they were coding for, generally a single device or possibly a range of devices from a single manufacturer. As hardware technology has advanced, this closed approach has struggled to keep pace.

More recently, platforms like Symbian have been created to provide developers a wider target audience. These systems have proved more successful in encouraging mobile developers to provide rich applications that better leverage the hardware available.

These platforms offer some access to the device hardware, but require writing complex C/C++ code and making heavy use of proprietary APIs that are notoriously difficult to use. This difficulty is amplified when developing applications that must work on different hardware implementations and is particularly true when developing for a particular hardware feature like GPS.

In recent years, the biggest advance in mobile phone development has been the introduction of Java-hosted MIDlets. MIDlets are executed on a Java virtual machine, abstracting the underlying hardware and letting developers create applications that run on the wide variety of hardware that supports the Java run time. Unfortunately, this convenience comes at the price of restricted access to the device hardware.

In mobile development, it's considered normal for third-party applications to receive different hardware access and execution rights compared to native applications written by the phone manufacturers, with MIDlets often receiving few of either.

The introduction of Java MIDlets has expanded developers' audiences, but the lack of low-level hardware access and sandboxed execution have meant that most mobile applications are desktop programs designed to run on a smaller screen rather than take advantage of the inherent mobility of the handheld platform.

The Future

Android sits alongside a new wave of mobile operating systems designed for increasingly powerful mobile hardware. Windows Mobile and Apple's iPhone now provide a richer, simplified development environment for mobile applications. However, unlike Android, they're built on proprietary operating systems that often prioritize native applications over those created by third parties and restrict communication among applications and native phone data. Android offers new possibilities for mobile applications by offering an open development environment built on an open source Linux kernel. Hardware access is available to all applications through a series of API libraries, and application interaction, while carefully controlled, is fully supported.

In Android, all applications have equal standing. Third-party and native Android applications are written using the same APIs and are executed on the same run time. Users can remove and replace any native application with a third-party developer alternative; even the dialer and home screens can be replaced.

What It Isn't

As a disruptive addition to a mature field, it's not hard to see why there has been some confusion about what exactly Android is. Android is **not**:

- **A Java ME implementation** Android applications are written using the Java language, but they are not run within a Java ME virtual machine, and Java-compiled classes and executables will not run natively in Android.
- **Part of the Linux Phone Standards Forum (LiPS) or the Open Mobile Alliance (OMA)** Android runs on an open source Linux kernel, but, while their goals are similar, Android's complete software stack approach goes further than the focus of these standards-defining organizations.
- **Simply an application layer (like UIQ or S60)** While it does include an application layer, "Android" also describes the entire software stack encompassing the underlying operating system, API libraries, and the applications themselves.
- **A mobile phone handset** Android includes a reference design for mobile handset manufacturers, but unlike the iPhone, there is no single "Android Phone." Instead, Android has been designed to support many alternative hardware devices.
- **Google's answer to the iPhone** The iPhone is a fully proprietary hardware and software platform released by a single company (Apple), while Android is an open source software stack produced and supported by the Open Handset Alliance and designed to operate on any handset that meets the requirements. There's been a lot of speculation regarding a Google-branded Android phone, but even should Google produce one, it will be just one company's hardware implementation of the Android platform.

An Open Platform for Mobile Development

Google describes Android as:

> *The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.*
>
> `http://googleblog.blogspot.com/2007/11/wheres-my-gphone.html`

Android is made up of several necessary and dependent parts including the following:

- A hardware reference design that describes the capabilities required of a mobile device in order to support the software stack
- A Linux operating system kernel that provides the low-level interface with the hardware, memory management, and process control, all optimized for mobile devices
- Open source libraries for application development including SQLite, WebKit, OpenGL, and a media manager
- A run time used to execute and host Android applications, including the Dalvik virtual machine and the core libraries that provide Android specific functionality. The run time is designed to be small and efficient for use on mobile devices.
- An application framework that agnostically exposes system services to the application layer, including the window manager, content providers, location manager, telephony, and peer-to-peer services
- A user interface framework used to host and launch applications
- Preinstalled applications shipped as part of the stack
- A software development kit used to create applications, including the tools, plug-ins, and documentation

At this stage, not all of the Android stack has been released as open source, although this is expected to happen by the time phones are released to market. It's also worth noting that the applications you develop for Android do not have to be open source.

What really makes Android compelling is its open philosophy, which ensures that any deficiencies in user interface or native application design can be fixed by writing an extension or replacement. Android provides you, as a developer, the opportunity to create mobile phone interfaces and applications designed to look, feel, and function exactly as you image them.

Native Android Applications

Android phones will normally come with a suite of preinstalled applications including, but not limited to:

- An e-mail client compatible with Gmail but not limited to it
- An SMS management application
- A full PIM (personal information management) suite including a calendar and contacts list, both tightly integrated with Google's online services

- A fully featured mobile Google Maps application including StreetView, business finder, driving directions, satellite view, and traffic conditions
- A WebKit-based web browser
- An Instant Messaging Client
- A music player and picture viewer
- The Android Marketplace client for downloading thied-party Android applications.
- The Amazon MP3 store client for purchasing DRM free music.

All the native applications are written in Java using the Android SDK and are run on Dalvik.

The data stored and used by the native applications — like contact details — are also available to third-party applications. Similarly, your applications can handle events such as an incoming call or a new SMS message.

The exact makeup of the applications available on new Android phones is likely to vary based on the hardware manufacturer and/or the phone carrier or distributor. This is especially true in the United States, where carriers have significant influence on the software included on shipped devices.

Android SDK Features

The true appeal of Android as a development environment lies in the APIs it provides.

As an application-neutral platform, Android gives you the opportunity to create applications that are as much a part of the phone as anything provided out of the box. The following list highlights some of the most noteworthy Android features:

- No licensing, distribution, or development fees
- Wi-Fi hardware access
- GSM, EDGE, and 3G networks for telephony or data transfer, allowing you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks
- Comprehensive APIs for location-based services such as GPS
- Full multimedia hardware control including playback and recording using the camera and microphone
- APIs for accelerometer and compass hardware
- IPC message passing
- Shared data stores
- An integrated open source WebKit-based browser
- Full support for applications that integrate Map controls as part of their user interface
- Peer-to-peer (P2P) support using Google Talk
- Mobile-optimized hardware-accelerated graphics including a path-based 2D graphics library and support for 3D graphics using OpenGL ES

- Media libraries for playing and recording a variety of audio/video or still image formats
- An application framework that encourages reuse of application components and the replacement of native applications

Access to Hardware including Camera, GPS, and Accelerometer

Android includes API libraries to simplify development involving the device hardware. These ensure that you don't need to create specific implementations of your software for different devices, so you can create Android applications that work as expected on any device that supports the Android software stack.

The Android SDK includes APIs for location-based hardware (such as GPS), camera, network connections, Wi-Fi, Bluetooth, accelerometers, touch screen, and power management. You can explore the possibilities of some of Android's hardware APIs in more detail in Chapter 10.

Native Google Maps, Geocoding, and Location-Based Services

Native map support lets you create a range of map-based applications that leverage the mobility of Android devices. Android lets you create activities that include interactive Google Maps as part of your user interface with full access to maps that you can control programmatically and annotate using Android's rich graphics library.

Android's location-based services manage technologies like GPS and Google's GSM cell-based location technology to determine the device's current position. These services enforce an abstraction from specific location-detecting technology and let you specify minimum requirements (e.g., accuracy or cost) rather than choosing a particular technology. It also means that your location-based applications will work no matter what technology the host handset supports.

To combine maps with locations, Android includes an API for forward and reverse geocoding that lets you find map coordinates for an address, and the address of a map position.

You'll learn the details of using maps, the geocoder, and location-based services in Chapter 7.

Background Services

Android supports applications and services designed to run invisibly in the background.

Modern mobiles are by nature multifunction devices; however, their limited screen size means that generally only one interactive application can be visible at any time. Platforms that don't support background execution limit the viability of applications that don't need your constant attention.

Background services make it possible to create invisible application components that perform automatic processing without direct user action. Background execution allows your applications to become event-driven and to support regular updates, which is perfect for monitoring game scores or market prices, generating location-based alerts, or prioritizing and pre-screening incoming calls and SMS messages.

Learn more about how to get the most out of background services in Chapter 8.

SQLite Database for Data Storage and Retrieval

Rapid and efficient data storage and retrieval are essential for a device whose storage capacity is limited by its compact nature.

Android provides a lightweight relational database for each application using SQLite. Your applications can take advantage of the managed relational database engine to store data securely and efficiently.

By default, each application database is *sandboxed* — its content is available only to the application that created it — but Content Providers supply a mechanism for the managed sharing of these application databases.

Databases, Content Providers, and other data persistence options available in Android are covered in detail in Chapter 6.

Shared Data and Interapplication Communication

Android includes three techniques for transmitting information from your applications for use elsewhere: Notifications, Intents, and Content Providers.

Notifications are the standard ways in which a mobile device traditionally alerts users. Using the API, you can trigger audible alerts, cause vibration, and flash the device's LED, as well as control status bar notification icons as shown in Chapter 8.

Intents provide a mechanism for message passing within and between applications. Using Intents, you can broadcast a desired action (such as dialing the phone or editing a contact) system-wide for other applications to handle. Intents are an important core component of Android and are covered in depth in Chapter 5.

Finally, *Content Providers* are a way to give managed access to your application's private database. The data stores for native applications, such as the Contact Manager, are exposed as Content Providers so you can create your own applications that read or modify these data stores. Chapter 6 covers Content Providers in detail, including the native providers and demonstrating how to create and use providers of your own.

P2P Services with Google Talk

Based on earlier SDK versions, it's expected that in later releases you will once again be able to send structured messages from your application to any other Android mobile using Android's peer-to-peer (P2P) communications service.

The Android P2P service uses a specialized version of XMPP (Extensible Messaging and Presence Protocol). Based on Google's Google Talk instant messaging service, it creates a persistent socket connection between your device and any other online Android handset that ensures communication with low latency and rapid response times.

When made available, you'll be able to use the Google Talk service for conventional instant messaging, or an interface to send data between application instances on separate devices. This is strong sauce for creating interactive applications that involve multiple users, such as real-time multiplayer games or social applications.

The P2P service also offers presence notification, which is used to see if a contact is online. While the P2P service is very attractive in itself, it also plays very well with other Android features. Imagine a background service that transmits locations between friends and a corresponding mapping application that displays these locations or alerts you when friends are nearby.

Owing to security concerns, sending data messages with Google Talk isn't possible in Android 1.0. An instant messaging client is available, and it's expected that XMPP-compatible IM and data messaging will be made available to developers in a future SDK release.

Extensive Media Support and 2D/3D Graphics

Bigger screens and brighter, higher-resolution displays have helped make mobiles multimedia devices. To make the most of the hardware available, Android provides graphics libraries for 2D canvas drawing and 3D graphics with OpenGL.

Android also offers comprehensive libraries for handling still images, video, and audio files including the MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, and GIF formats.

2D and 3D graphics are covered in depth in Chapter 11, while Android media management libraries are covered in Chapter 10.

Optimized Memory and Process Management

Android's process and memory management is a little unusual. Like Java and .NET, Android uses its own run time and virtual machine to manage application memory. Unlike either of these frameworks, the Android run time also manages the process lifetimes. Android ensures application responsiveness by stopping and killing processes as necessary to free resources for higher-priority applications.

In this context, priority is determined depending on the application with which the user is interacting. Ensuring that your applications are prepared for a swift death but are still able to remain responsive and update or restart in the background if necessary, is an important consideration in an environment that does not allow applications to control their own lifetimes.

You will learn more about the Android application life cycle in Chapter 3.

Introducing the Open Handset Alliance

The *Open Handset Alliance (OHA)* is a collection of more than 30 technology companies including hardware manufacturers, mobile carriers, and software developers. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm. In their own words, the OHA represents:

> *A commitment to openness, a shared vision for the future, and concrete plans to make the vision a reality. To accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.*
>
> `http://www.openhandsetalliance.com/oha_faq.html`

The OHA hopes to deliver a better mobile software experience for consumers by providing the platform needed for innovative mobile development at a faster rate and a higher quality without licensing fees for software developers or handset manufacturers.

Ultimately the success of Android as a mobile platform will depend largely on the success of OHA partners in releasing desirable handsets and mobile services that encourage the widespread adoption of Android phones. Developers meanwhile have the opportunity to create innovative new mobile applications for Android to encourage more mobile technology companies to become part of the OHA.

What Does Android Run On?

The first Android mobile handset, the T-Mobile G1, was released in the US in October 2008 and in the UK in November 2008. The Open Handset Alliance has further committed to deploying additional handsets and services that support Android early in 2009.

Rather than a mobile OS created for a single hardware implementation, Android is designed to support a large variety of hardware platforms, from touch-screen phones to devices with no screens at all.

Beyond that, with no licensing fees or proprietary software, the cost to handset manufacturers for providing Android-compatible variations of their handsets is comparatively low. It's hoped that once demand for hardware capable of running popular Android applications reaches a critical mass, more device manufacturers will produce increasingly tailored hardware to meet that demand.

Why Develop for Android?

If you have a background in mobile application development, you don't need me to tell you that:

- A lot of what you can do with Android is already possible.
- But doing it is painful.

Android represents a clean break, a mobile framework based on the reality of modern mobile devices.

With a simple and powerful SDK, no licensing fees, excellent documentation, and a thriving developer community, Android is an excellent opportunity to create software that changes how and why people use their mobile phones.

Android is backed by more than 30 OHA members and is surrounded by significant industry buzz.

In market terms, the growth in portable devices is a worldwide phenomenon, with mobile-phone ownership outstripping computer ownership in many countries. The increasing popularity of *smartphones* — multifunction devices including a phone but featuring cameras, Internet access, media players, Wi-Fi, and GPS services — combined with the increasing availability of mobile broadband and Wi-Fi has created a growth market for advanced mobile applications.

What Will Drive Android Adoption?

Android is targeted primarily at developers, with Google and the OHA betting that the way to deliver better mobile software to consumers is by making it easier for developers to write it themselves.

As a development platform, Android is powerful and intuitive, letting developers who have never programmed for mobile devices create useful applications quickly and easily. It's easy to see how innovative Android applications could create demand for the devices necessary to run them, particularly if developers write applications for Android because they *can't* write them for other platforms.

Open access to the nuts and bolts of the underlying system is what's always driven software development and platform adoption. The Internet's inherent openness and neutrality have seen it become the platform for a multi-billion-dollar industry within 10 years of its inception. Before that, it was open systems like Linux and the powerful APIs provided as part of the Windows operating system that enabled the explosion in personal computers and the movement of computer programming from the arcane to the mainstream.

This openness and power ensure that anyone with the inclination can bring a vision to life at minimal cost. So far, that's not been the case for mobile phones, and that's why there are so few good mobile phone applications and fewer still available for free.

Corporations will also be attracted to Android for the level of control it offers. By using a popular enterprise programming language in Java, no licensing fees, and offering the level of access and control users demand, Android offers an excellent enterprise platform.

What Does It Have That Others Don't?

Many of the features listed previously, such as 3D graphics and native database support, are also available in other mobile SDKs. Here are some of the unique features that set Android apart:

- **Google Map Application**s Google Maps for Mobile has been hugely popular, and Android offers a Google Map as an atomic, reusable control for use in your applications. The MapView widget lets you display, manipulate, and annotate a Google Map within your Activities to build map-based applications using the familiar Google Maps interface.
- **Background Services and Application**s Background services let you create applications that use an event-driven model, working silently while other applications are being used or while your mobile sits ignored until it rings, flashes, or vibrates to get your attention. Maybe it's an application that tracks the stock market, alerting you to significant changes in your portfolio, or a service that changes your ring tone or volume depending on your current location, the time of day, and the identity of the caller.

- **Shared Data and Interprocess Communication** Using Intents and Content Providers, Android lets your applications exchange messages, perform processing, and share data. You can also use these mechanisms to leverage the data and functionality provided by the native Android applications. To mitigate the risks of such an open strategy, each application's process, data storage, and files are private unless explicitly shared with other applications using a full permission-based security mechanism detailed in Chapter 11.
- **All Applications Are Created Equal** Android doesn't differentiate between native applications and those developed by third parties. This gives consumers unprecedented power to change the look and feel of their devices by letting them completely replace every native application with a third-party alternative that has access to the same underlying data and hardware. Every rule needs an exception and this one has two. The "unlock" and "in-call experience" screens can not be replaced in the initial SDK release.
- **P2P Interdevice Application Messaging** Android offers peer-to-peer messaging that supports presence, instant messaging, and interdevice/interapplication communication.

Changing the Mobile Development Landscape

Existing mobile development platforms have created an aura of exclusivity around mobile development. Whether by design or as a side-effect of the cost or complexity involved in developing native applications, most mobile phones will remain nearly identical to what they were when first unwrapped.

In contrast, Android allows, even encourages, radical change. As consumer devices, Android handsets ship with a core set of standard applications that consumers demand on a new phone, but the real power lies in the ability for users to completely change how their device looks, feels, and functions.

Android gives developers a great opportunity. All Android applications are a native part of the phone, not just software that's run in a sandbox on top of it. Rather than writing small-screen versions of software that can be run on low-power devices, you can now write mobile applications that change the way people use their phones.

While Android will still have to compete with existing and future mobile development platforms as an open source developer framework, the strength of use of the development environment is strongly in its favor. Certainly its free and open approach to mobile application development, with total access to the phone's resources, is a giant step in the right direction.

Introducing the Development Framework

With the PR job done, it's time to look at how you can start developing applications for Android. Android applications are written using Java as a programming language but are executed using a custom virtual machine called *Dalvik* rather than a traditional Java VM.

Later in this chapter, you'll be introduced to the framework, starting with a technical explanation of the Android software stack, a look at what's included in the SDK, an introduction to the Android libraries, and a look at the Dalvik virtual machine.

Each Android application runs in a separate process within its own Dalvik instance, relinquishing all responsibility for memory and process management to the Android run time, which stops and kills processes as necessary to manage resources.

Dalvik and the Android run time sit on top of a Linux kernel that handles low-level hardware interaction including drivers and memory management, while a set of APIs provides access to all of the underlying services, features, and hardware.

What Comes in the Box

The Android software development kit (SDK) includes everything you need to start developing, testing, and debugging Android applications. Included in the SDK download are:

- **The Android APIs** The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries used at Google to create native Android applications.
- **Development Tools** To turn Android source code into executable Android applications, the SDK includes several development tools that let you compile and debug your applications. You will learn more about the developer tools in Chapter 2.
- **The Android Emulator** The Android Emulator is a fully interactive Android device emulator featuring several alternative skins. Using the emulator, you can see how your applications will look and behave on a real Android device. All Android applications run within the Dalvik VM so that the software emulator is an excellent environment — in fact, as it is hardware-neutral, it provides a better independent test environment than any single hardware implementation.
- **Full Documentation** The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them. In addition to the code documentation, Android's reference documentation explains how to get started and gives detailed explanations of the fundamentals behind Android development.
- **Sample Code** The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available using Android, as well as simple programs that highlight how to use individual API features.
- **Online Support** Despite its relative youth, Android has generated a vibrant developer community. The Google Groups at `http://code.google.com/android/groups` are active forums of Android developers with regular input from the Android development team at Google.

For those using the popular Eclipse IDE, Android has released a special plug-in that simplifies project creation and tightly integrates Eclipse with the Android Emulator and debugging tools. The features of the ADT plug-in are covered in more detail in Chapter 2.

Understanding the Android Software Stack

The Android software stack is composed of the elements shown in Figure 1-1 and described in further detail below it. Put simply, a Linux kernel and a collection of C/C++ libraries are exposed through an application framework that provides services for, and management of, the run time and applications.

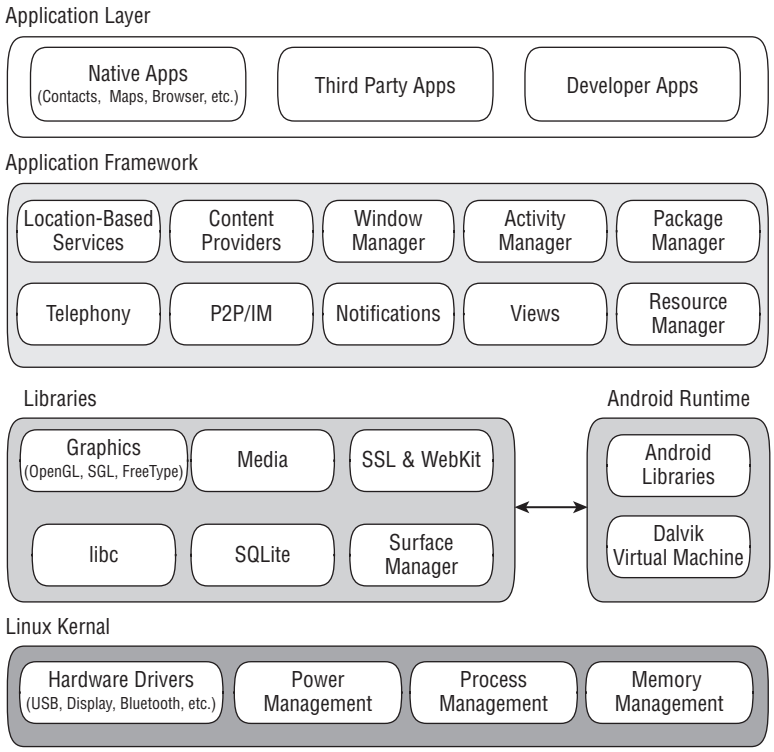

Figure 1-1

- **Linux Kernel** Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.
- **Libraries** Running on top of the kernel, Android includes various C/C++ core libraries such as libc and SSL, as well as:
  - A media library for playback of audio and video media
  - A Surface manager to provide display management
  - Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
  - SQLite for native database support
  - SSL and WebKit for integrated web browser and Internet security
- **Android Run Time** What makes an Android phone an Android phone rather than a mobile Linux implementation is the Android run time. Including the core libraries and the Dalvik virtual machine, the Android run time is the engine that powers your applications and, along with the libraries, forms the basis for the application framework.
  - **Core Libraries** While Android development is done in Java, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android-specific libraries.

    - **Dalvik Virtual Machine** Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

- **Application Framework** The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- **Application Layer** All applications, both native and third party, are built on the application layer using the same API libraries. The application layer runs within the Android run time using the classes and services made available from the application framework.

The Dalvik Virtual Machine

One of the key elements of Android is the Dalvik virtual machine. Rather than use a traditional Java virtual machine (VM) such as Java ME (Java Mobile Edition), Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.

The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management. It's also possible to write C/C++ applications that run directly on the underlying Linux OS. While you *can* do this, in most cases there's no reason you should need to.

> *This book focuses exclusively on writing applications that run within Dalvik. If your inclinations run toward exploring the Linux kernel and C/C++ underbelly of Android, modifying Dalvik, or otherwise tinkering with things under the hood, check out the Android Internals Google Group at* `http://groups.google.com/group/android-internals`

All Android hardware and system service access is managed using Dalvik as a middle tier. By using a VM to host application execution, developers have an abstraction layer that ensures they never have to worry about a particular hardware implementation.

The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory footprint. The `.dex` executables are created by transforming Java language compiled classes using the tools supplied within the SDK. You'll learn more about how to create Dalvik executables in the next chapter.

Android Application Architecture

Android's architecture encourages the concept of component reuse, allowing you to publish and share activities, services, and data with other applications with access managed by the security restrictions you put in place.

The same mechanism that lets you produce a replacement contact manager or phone dialer can let you expose your application components to let other developers create new UI front ends and functionality extensions, or otherwise build on them.

The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:

- **Activity Manager** Controls the life cycle of your activities, including management of the activity stack described in Chapter 3.
- **Views** Are used to construct the user interfaces for your activities as described in Chapter 4.
- **Notification Manager** Provides a consistent and non-intrusive mechanism for signaling your users as described in Chapter 8.
- **Content Providers** Lets your applications share data between applications as described in Chapter 6.
- **Resource Manager** Supports non-code resources like strings and graphics to be externalized as shown in Chapter 3.

Android Libraries

Android offers a number of APIs for developing your applications. The following list of core APIs should provide an insight into what's available; all Android devices will offer support for at least these APIs:

- **android.util** The core utility package contains low-level classes like specialized containers, string formatters, and XML parsing utilities.
- **android.os** The operating system package provides access to basic operating system services like message passing, interprocess communication, clock functions, and debugging.
- **android.graphics** The graphics API supplies the low-level graphics classes that support canvases, colors, and drawing primitives, and lets you draw on canvases.
- **android.text** The text processing tools for displaying and parsing text.
- **android.database** Supplies the low-level classes required for handling cursors when working with databases.
- **android.content** The content API is used to manage data access and publishing by providing services for dealing with resources, content providers, and packages.
- **android.view** Views are the core user interface class. All user interface elements are constructed using a series of Views to provide the user interaction components.
- **android.widget** Built on the View package, the widget classes are the "here's one we created earlier" user-interface elements for you to use in your applications. They include lists, buttons, and layouts.
- **com.google.android.maps** A high-level API that provides access to native map controls that you can use within your application. Includes the MapView control as well as the Overlay and MapController classes used to annotate and control your embedded maps.

- **android.app** A high-level package that provides access to the application model. The application package includes the Activity and Service APIs that form the basis for all your Android applications.
- **android.provider** To ease developer access to certain standard Content Providers (such as the contacts database), the Provider package offers classes to provide access to standard databases included in all Android distributions.
- **android.telephony** The telephony APIs give you the ability to directly interact with the device's phone stack, letting you make, receive, and monitor phone calls, phone status, and SMS messages.
- **android.webkit** The WebKit package features APIs for working with Web-based content, including a WebView control for embedding browsers in your activities and a cookie manager.

In addition to the Android APIs, the Android stack includes a set of C/C++ libraries that are exposed through the application framework. These libraries include:

- **OpenGL** The library used to support 3D graphics based on the Open GL ES 1.0 API
- **FreeType** Support for bitmap and vector font rendering
- **SGL** The core library used to provide a 2D graphics engine
- **libc** The standard C library optimized for Linux-based embedded devices
- **SQLite** The lightweight relation database engine used to store application data
- **SSL** Support for using the Secure Sockets Layer cryptographic protocol for secure Internet communications

Advanced Android Libraries

The core libraries provide all the functionality you need to start creating applications for Android, but it won't be long before you're ready to delve into the advanced APIs that offer the really exciting functionality.

Android hopes to target a wide range of mobile hardware, so be aware that the suitability and implementation of the following APIs will vary depending on the device upon which they are implemented.

- **android.location** The location-based services API gives your applications access to the device's current physical location. Location-based services provide generic access to location information using whatever position-fixing hardware or technology is available on the device.
- **android.media** The media APIs provide support for playback and recording of audio and video media files, including streamed media.
- **android.opengl** Android offers a powerful 3D rendering engine using the OpenGL ES API that you can use to create dynamic 3D user interfaces for your applications.
- **android.hardware** Where available, the hardware API exposes sensor hardware including the camera, accelerometer, and compass sensors as shown in Chapter 10.
- **android.bluetooth, android.net.wifi, and android.telephony** Android also provides low-level access to the hardware platform, including Bluetooth, Wi-Fi, and telephony hardware as shown in Chapter 10.

Summary

This chapter explained that despite significant advances in the hardware features available on modern mobile phones, the software available for them has lagged. A lack of openness, hard-to-use development kits, and hardware-specific APIs have stifled innovation in mobile software.

Android offers an opportunity for developers to create innovative software applications for mobile devices without the restrictions generally associated with the existing proprietary mobile development frameworks.

You were shown the complete Android software stack, which includes not only an application layer and development toolkit but also the Dalvik VM, a custom run time, core libraries, and a Linux kernel; all of which will be available as open source.

The Open Handset Alliance was introduced along with the responsibility that developers — as the primary target audience for Android — have to create applications that will make consumers want Android phones on which to run them.

You also learned:

- How handsets with an expanding range of hardware features have created demand for tools that give developers better access to these features.
- About some of the features available to developers using Android, including peer-to-peer messaging, native map support, hardware access, background services, interprocess and inter-device messaging, shared databases, and 2D and 3D graphics.
- That all Android applications are built equal, allowing users to completely replace one application with another, including the replacement of the core native applications.
- That the Android SDK includes developer tools, APIs, and comprehensive documentation.

The next chapter will help you get started by downloading and installing the Android SDK and setting up an Android development environment in Eclipse.

You'll also learn how to use the Android developer tools plug-in to streamline development, testing, and debugging before creating your first Android application.

After learning about the building blocks of Android applications, you'll be introduced to the different types of applications you can create, and you'll start to understand some of the design considerations that should go into developing applications for mobile devices.