Introduction

The Android platform is packing some serious heat these days in the mobile marketplace and gaining traction worldwide. The platform has seen numerous advancements in terms of SDK functionality, handset availability, and feature set. A wide diversity of Android handsets and devices are now shipping and (finally) in consumers’ hands—and we’re not just talking about phones: Android has begun to ship on netbooks, Internet tablets (such as the ARCHOS 5), ebook readers (like the Barnes & Noble nook), digital photo frames, and a variety of other consumer electronics. There are even proof-of-concept appliances such as an Android microwave and washer/dryer combo. (Hey, why not? See http://bit.ly/bGqmZp.) Mobile operators and carriers are taking the platform seriously and spending gazillions on ad campaigns for Android phones—like Verizon’s Droid campaign.

In the past year or so, the Android platform has transitioned from a “gearheads-only” platform to providing some serious competition to more established platforms. (Yes, we’re talking about platforms such as the iPhone.)

But let’s not digress into an argument over whose platform is better so early, okay? Because, honestly, you’re wasting your time if you think there’s one platform to rule them all. The reality is, people the world over use different phones in different places (CDMA, GSM) and for different reasons (price, availability, coverage quality, feature set, design, familiarity, compatibility). There is no one-size-fits-all answer to this debate.

Having developed for just about every major mobile platform out there, we are keenly aware of the benefits and drawbacks of each platform. We do not presume to claim that one platform is better than another in general; each platform has distinct advantages over the rest, and these advantages can be maximized.

The trick is to know which platform to use for a given project. Sometimes, the answer is to use as many platforms as possible. Lately, we’ve been finding that the answer is the Android platform: It’s inexpensive and easy to develop for, it’s available to millions of potential users worldwide, and it has fewer limitations than other platforms.

Still, the Android platform is relatively young and has not yet reached its full-fledged potential. This means frequent SDK updates, an explosion of new devices on the market, and a nearly full-time job keeping track of everything going on in the Android world.

In other words, it may be a bit of a bumpy ride, but there’s still time to jump on this bandwagon, write some kick-butt applications, and make a name for yourself.

So let’s get to it.
Who Should Read This Book?

There’s no reason anyone with an Android handset and a good idea for a mobile application couldn’t put this book to use for fun and profit. Whether you’re a programmer looking to break into mobile technology or an entrepreneur with a cool app idea, this book is for you.

We make very few assumptions about you as a reader of this book. You may have a basic understanding of the Java programming language (understanding classes, methods, basic inheritance, and so on), but Android makes a fantastic platform for learning Java as well. We have avoided using any fancy or confusing Java in this book, so if you’re just getting started with programming, you should be able to read the first few chapters of any introductory Java book or do an online tutorial and have enough Java knowledge to make it through this book alive.

We do assume that you’re somewhat comfortable installing applications on a computer (for example, Eclipse, the Java JDK, and the Android SDK) and tools and drivers (for USB access to a phone), and we assume that you can navigate your way around an Android handset well enough to launch applications and such. No wireless development experience is necessary.

How This Book Is Structured

In 24 easy one-hour lessons, you’ll design and develop a fully functional network- and LBS (Location-Based Services)-enabled Android application, complete with social features. Each lesson builds on your knowledge of newly introduced Android concepts, and you’ll iteratively improve your application from chapter to chapter.

This book is divided into six parts:

- **Part I: Android Fundamentals**
  In Part I, you’ll get an introduction to Android, become familiar with the Android SDK and tools, install the development tools, and write your first Android application. Part I also introduces the design principles necessary to write Android applications, including how Android applications are structured and configured, as well as how to incorporate application resources such as strings, graphics, and user interface components into your projects.

- **Part II: Building an Application Framework**
  In Part II, you’ll begin developing an application framework that will serve as primary teaching-tool for the rest of the book. You’ll start by developing an animated splash screen, followed by screens for main menu, settings, help, and scores. You’ll
learn basic user interface design principles, how to collect input from the user, and how to display dialogs to the user. Finally, you’ll implement the core application logic of the game screen.

Part III: Enhancing Your Application with Powerful Android Features

In Part III, you’ll dive deeper into the Android SDK, adding more specialized features to the Been There, Done That! application. You’ll learn how to work with graphics and the built-in camera, how to leverage LBS, how to network-enable your application, and how to enhance your application with social features.

Part IV: Adding Polish to Your Android Application

In Part IV, you’ll learn how to customize your application for different handsets, screen sizes, and foreign languages. You’ll also learn about different ways to test mobile applications.

Part V: Publishing Your Application

In Part V, you’ll learn what you need to do to prepare for and publish your Android applications to the Android Market.

Part VI: Appendixes

In Part VI, you’ll find several helpful references for setting up your Android development environment, using the Eclipse IDE, and accessing supplementary book materials, like the book websites and downloadable source code.

What Is (and Isn’t) in This Book

While we specifically targeted Android SDK Version 2.1 in this book, many of the examples were tested on handsets running a variety of Android SDK versions.

The Android SDK is updated very frequently (every few months). We kept this in mind when choosing which features of the SDK to highlight to ensure maximum forward and backward compatibility. When necessary, we point out areas where the Android SDK version affects the features and functionality available to the developer.

This book is written in a beginner’s tutorial style. If you’re looking for an exhaustive reference on Android development, with cookbook-style code examples and a more thorough examination of all the features of the Android platform, we recommend our other, more advanced Android book, *Android Wireless Application Development*, which is part of the Addison-Wesley Developer’s Library series.
What Development Environment Is Used?

The code in this book was written using the following development environments:

- Windows 7 and Mac OS X 10.6
- Eclipse Java IDE Version 3.5 (Galileo)
- Eclipse JDT plug-in and Web Tools Platform (WTP)
- Sun Java SE Development Kit (JDK) 6 Update 18
- Android SDK Version 2.1 (Primary target, developed and tested on a variety of SDK versions)
- Various Android handsets (Android SDK 1.6, 2.0.1, and 2.1)

What Conventions Are Used in This Book?

This book presents several types of sidebars for special kinds of information:

- **Did You Know?** messages provide useful information or hints related to the current text.
- **By the Way** messages provide additional information that might be interesting or relevant.
- **Watch Out!** messages provide hints or tips about pitfalls that may be encountered and how to avoid them.

This book uses the following code-related conventions:

- Code and programming terms are set in a monospace font.
- ➥ is used to signify that the code that follows should appear on the same line as the preceding code.
- Exception handling and error checking are often removed from printed code samples for clarity and to keep the book a reasonable length.

This book uses the following conventions for step-by-step instructions and explanations:

- The core application developed in this book is developed iteratively. Generally, this means that the first time a new concept is explained, every item related to the new concept is discussed in detail. As we move on to more advanced topics in later lessons,
we assume that you have mastered some of the more rudimentary aspects of Android development from previous chapters, and we do not repeat ourselves much. In some cases, we instruct you to implement something in an early lesson and then help you improve it in a later chapter.

We assume that you’ll read the chapters of this book in order. As you progress through the book, you’ll note that we do not spell out each and every step that must be taken for each and every feature you implement to follow along in building the core application example. For example, if three buttons must be implemented on a screen, we walk you step-by-step through the implementation of the first button but leave the implementation of the other two buttons as an exercise for you. In a later chapter on a different topic, we might simply ask you to implement some buttons on another screen.

Where we tell you to navigate through menu options, we separate options using commas. For example, if we told you to open a new document, we’d say “Select File, New Document.”
Android developers are lucky to have more than a dozen development tools at their disposal to help facilitate the design of quality applications. Understanding what tools are available and what they can be used for is a task best done early in the Android learning process, so that when you are faced with a problem, you have some clue as to which utility might be able to help you find a solution. The Android development tools are found in the /tools subdirectory of the Android SDK installation. During this hour, we walk through a number of the most important tools available for use with Android. This information will help you develop Android applications faster and with fewer roadblocks.

Using the Android Documentation

Although it is not a tool, per se, the Android documentation is a key resource for Android developers. An HTML version of the Android documentation is provided in the /docs subfolder of the Android SDK documentation, and this should always be your first stop when you encounter a problem. You can also access the latest help documentation online at the Android Developer website, http://developer.android.com.

The Android documentation is divided into six sections (see Figure 2.1):
SDK—This tab provides important information about the SDK version installed on your machine. One of the most important features of this tab is the release notes, which describe any known issues for the specific installation. This information is also useful if the online help has been upgraded but you want to develop to an older version of the SDK.

Dev Guide—This tab links to the Android Developer’s Guide, which includes a number of FAQs for developers, as well as step-by-step examples and a useful glossary of Android terminology for those new to the platform.

Reference—This tab includes a searchable package and class index of all Android APIs provided as part of the Android SDK.

Blog—This tab links to the official Android developer blog. Check here for the latest news and announcements about the Android platform. This is a great place to find how-to examples, learn how to optimize Android applications, and hear about new SDK releases and Android Developer Challenges.

Videos—This tab, which is available online only, is your resource for Android training videos. Here, you’ll find videos about the Android platform, developer tips, and the Google I/O conference sessions.

Community—This tab is your gateway to the Android developer forums. There are a number of Google groups you can join, depending on your interests.

FIGURE 2.1
Android developer documentation (online version).
Now is a good time to get to know your way around the Android SDK documentation. First, try the local documentation and then check out the online documentation.

**Debugging Applications with DDMS**

The Dalvik Debug Monitor Service (DDMS) is a debugging utility that is integrated into Eclipse through the DDMS perspective. The DDMS perspective provides a number of useful features for interacting with emulators and handsets (see Figure 2.2).

The features of DDMS are roughly divided into five functional areas:

- Task management
- File management
- Emulator interaction
- Logging
- Screen captures
Did you Know?

The DDMS tool can be launched separately from Eclipse. You can find it in the Android SDK /tools directory.

Managing Tasks

The top-left corner of the DDMS lists the emulators and handsets currently connected. You can select individual instances and inspect processes and threads. You can inspect threads by clicking on the device process you are interested in—for example, com.androidbook.droid1—and clicking the Update Threads button ( ), as shown in Figure 2.3. You can also prompt garbage collection on a process and then view the heap updates by clicking the green cylinder button ( ). Finally, you can stop a process by clicking the button that resembles a stop sign ( ).

Debugging from the DDMS Perspective

Within the DDMS perspective, you can choose a specific process on an emulator or a handset and then click the little green bug ( ) to attach a debugger to that process. You need to have the source code in your Eclipse workspace for this to work properly. This works only in Eclipse, not in the standalone version of DDMS.
Browsing the Android File System

You can use the DDMS File Explorer to browse files and directories on the emulator or a device (see Figure 2.4). You can copy files between the Android file system and your development machine by using the push (.JTextField) and pull ( JTextField) icons.

You can also delete files and directories by using the minus button ( JTextField) or just pressing Delete. There is no confirmation for this Delete operation, nor can it be undone.

Interacting with Emulators

DDMS can send a number of events, such as simulated calls, SMS messages, and location coordinates, to specific emulator instances. These features are found under the Emulator Control tab in DDMS. These events are all “one way,” meaning that they can be initiated from DDMS, not from the emulator to DDMS.

Simulating Incoming Calls to the Emulator

You can simulate incoming voice calls by using the DDMS Emulator Control tab (see Figure 2.5). This is not a real call; no data (voice or otherwise) is transmitted between the caller and the receiver.
HOUR 2: Mastering the Android Development Tools

FIGURE 2.5
Using the DDMS Emulator Control tab (left) to place a call to the emulator (right).

Try It Yourself

Simulate an Incoming Call to an Emulator

To simulate an incoming call to an emulator running on your machine, follow these steps:

1. In DDMS, choose the emulator you want to call.
2. On the Emulator Control tab, input the incoming phone number (for example, 5551212) in the Telephony Actions section.
3. Select the Voice radio button.
4. Click the Call button.
5. In the emulator, you should see an incoming call. Answer the call by clicking the Send button in the emulator.
6. End the call at any time by clicking the End button in the emulator or by clicking the Hang Up button on the DDMS Emulator Control tab.
Simulating Incoming SMS Messages to the Emulator

You can simulate incoming SMS messages by using the DDMS Emulator DDMS (see Figure 2.6). You send an SMS much as you initiate a voice call.

![Using the DDMS Emulator Control tab (left) to send an SMS message to the emulator (right).]

**Try It Yourself**

Send an SMS to the Emulator

To send an SMS message to an emulator running on your machine, follow these steps:

1. In DDMS, choose the emulator you want a send an SMS to.
2. On the Emulator Control tab, input the Incoming phone number (for example, 5551212) in the Telephony Actions section.
3. Select the SMS radio button.
4. Type an SMS message.
5. Click the Send button. In the emulator, you should see an incoming SMS notification.

Taking Screenshots of the Emulator or Handset

One feature that can be particularly useful for debugging both handsets and emulators is the ability to take screenshots of the current screen (see Figure 2.7).
FIGURE 2.7
Using the DDMS Screen Capture button to take a screenshot of the handset.

Try It Yourself

Take a Screen Capture

The screenshot feature is particularly useful when used with true handsets. To take a screen capture, follow these steps:

1. In DDMS, choose the device (or emulator) you want a screenshot of.
2. On that device or emulator, make sure you have the screen you want. Navigate to it, if necessary.
3. Choose the multicolored square picture icon to take a screen capture. This launches a capture screen dialog.
4. Within the capture screen, click Save to save the screenshot to your local hard drive.
**Viewing Log Information**

The LogCat logging utility that is integrated into DDMS allows you to view the Android logging console. You may have noted the LogCat logging tab, with its diagnostic output, in many of the figures shown so far in this chapter. We will talk more about how to implement your own custom application logging in Hour 3, “Building Android Applications.”

**Filtering Log Information**

Eclipse has the ability to filter logs by log severity. You can also create custom log filters by using tags. For more information on how to do this, see Appendix B, “Eclipse IDE Tips and Tricks.”

**Working with the Android Emulator**

The Android emulator is probably the most powerful tool at a developer’s disposal. It is important for developers to learn to use the emulator and understand its limitations.

The Android emulator is integrated with Eclipse, using the ADT plug-in for the Eclipse IDE.

**Emulator Limitations**

The Android emulator is a convenient tool, but it has a number of limitations:

- The emulator is not a device. It simulates general handset behavior, not specific hardware implementations.
- Sensor data, such as satellite location information, battery and power settings, and network connectivity, are all simulated using your computer.
- Peripherals such as camera hardware are not fully functional.
- Phone calls cannot be placed or received but are simulated. SMS messages are also simulated and do not use a real network.
- No USB or Bluetooth support is available.

Using Android emulator is not a substitute for testing on a true target handset or device.
Providing Input to the Emulator

As a developer, you can provide input to the emulator in a number of ways:

- Use your computer mouse to click, scroll, and drag items (for example, side volume controls) onscreen as well as on the emulator skin.
- Use your computer keyboard to input text into controls.
- Use your mouse to simulate individual finger presses on the soft keyboard or physical emulator keyboard.
- Use a number of emulator keyboard commands to control specific emulator states.

Try It Yourself

Try out some of the methods of interacting with the emulator:

1. In Eclipse, launch the Droid1 application you created in Hour 1, “Getting Started with Android.”

2. While your application is running, press the control-F11 and control-F12 keys to toggle the emulator orientation. Note how your application redraws the simple screen in portrait and landscape modes.

3. Press Alt+Enter to enter full screen mode with the emulator. Then press Alt+Enter again to return to normal mode.

Many useful commands are available. For an exhaustive list, see the official emulator documentation that was installed with the Android SDK documentation and is also available online, at http://developer.android.com/guide/developing/tools/emulator.html.

Exploring the Android System

If you’re not already familiar with Android devices, now is a good time to learn your way around the Android system as users see it. Table 2.1 lists some important features of Android.
### TABLE 2.1  Android System Screens and Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home screen</td>
<td>Default screen. This is a common location for app widgets and live folders.</td>
<td><img src="image" alt="Home screen" /></td>
</tr>
<tr>
<td>Dialer application</td>
<td>Built-in application for making and receiving phone calls. Note that the emulator has limited phone features.</td>
<td><img src="image" alt="Dialer" /></td>
</tr>
<tr>
<td>Messaging application</td>
<td>Built-in application for sending and receiving SMS messages. Note that the emulator has limited messaging features.</td>
<td><img src="image" alt="Messaging" /></td>
</tr>
<tr>
<td>Browser application</td>
<td>Built-in web browser. Note that the emulator has an Internet connection, provided that your machine has one.</td>
<td><img src="image" alt="Browser" /></td>
</tr>
<tr>
<td>Contacts application</td>
<td>Database of contact information.</td>
<td><img src="image" alt="Contacts" /></td>
</tr>
<tr>
<td>Application sliding drawer</td>
<td>Shows all installed applications. From the Home screen, pull the gray sliding drawer tab to see all installed applications.</td>
<td><img src="image" alt="Application sliding drawer" /></td>
</tr>
</tbody>
</table>
TABLE 2.1  continued

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings application</td>
<td>Built-in application to configure a wide variety of “phone” settings for the emulator, such as application management, sound and display settings, and localization.</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Dev Tools application</td>
<td>Built-in application to configure development tool settings.</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Using Emulator Skins

Emulator features such as screen size, screen orientation, and whether the emulator has a hardware or soft keyboard are dictated by the emulator skin. The Android SDK supports a number of different skins which emulate various handset screen resolutions (the default being HVGA). The specific skins available depends on the target build platform. Determining the appropriate skin is part of the AVD configuration process.

Using SD Card Images with the Emulator

To save data with the emulator, there must be an SD card image configured. For example, you must have a properly configured SD card image to save media files like camera graphics and sound files to the emulator.

The most convenient way to create SD card images for use with the emulator is to create them as part of the AVD process, as you did in Hour 1. SD card images should be at least 9 MiB.

Using Other Android Tools

Although we’ve already covered the most important tools, a number of other special-purpose utilities are included with the Android SDK:

- **Android Hierarchy Viewer**—Allows developers to inspect application user interface components such as View Properties while the application is running.
Summary

- **Draw 9-Patch tool**—Helps developers design stretchable PNG files.
- **AIDL Compiler**—Helps developers create remote interfaces to facilitate inter-process communication (IPC) on the Android platform.
- **mksdcard command-line utility**—Allows developers to create stand-alone SD card images for use within AVDs and the emulator.

---

**Developing Android Applications Without Eclipse**

Eclipse is the preferred development environment for Android, but it is not required for development. The ADT plug-in for Eclipse provides a convenient entry point for many of the underlying development tools for creating, debugging, packaging, and signing Android applications.

Developers who do not use Eclipse or require some of the more powerful debugging features not available in the Eclipse ADT plug-in can access these underlying tools directly from the command line. Tools such as the following are found in the /tools directory of the Android SDK:

- `android`—Creates Android project files and to manage AVDs.
- `aapt (Android Asset Packaging Tool)`—Packages Android project files into .apk files for installation on the emulator and handset.
- `ddms (Dalvik Debug Monitor Service)`—Has a user interface of its own, which resembles the Eclipse DDMS perspective.
- `adb (Android Debug Bridge)`—Has a command-line interface for interacting with the emulator and the device.

---

**Summary**

The Android SDK ships with a number of powerful tools to help with common Android development tasks. The Android documentation is an essential reference for developers. The DDMS debugging tool, which is integrated into the Eclipse development environment, is useful for monitoring emulators and devices. The Android emulator can be used for running and debugging Android applications virtually, without the need for an actual device. There are also a number of other tools for interacting with handsets and emulators at the command-line level, as well as specialized utilities for designing Android application user interfaces and graphics, as well as packaging applications.
Q&A

Q. Is the Android documentation installed with the Android SDK the same as the documentation found at http://developer.android.com?

A. No. The documentation installed with the SDK was “frozen” at the time the SDK was released, which means it is more specific to the version of the Android SDK you installed. The online documentation will always be the latest version of the Android SDK. We recommend using the online documentation, unless you are working offline or have a slow Internet connection, in which case the local SDK documentation will suffice.

Q. Do the different emulator skins have different features?

A. Yes. The emulator skins correspond to different screen sizes and orientations. They also have keypads and buttons. Some have hardware keyboards, and others rely on soft keyboard support.

Q. Is testing your application on the emulator alone sufficient?

A. No. The Android emulator simulates the functionality of a real device and can be a big time- and cost-saving tool for Android projects. It is a convenient tool for testing, but it can only pretend at real device behavior. The emulator cannot actually determine your real location or make a phone call. Also, the emulator is a generic device and does not attempt to simulate any quirky details of a specific handset. Just because your application runs fine on the emulator does not guarantee that it will work on the device.

Workshop

Quiz

1. Which features are available in the DDMS perspective?

A. Taking screenshots of emulator and handset screens

B. Browsing the file system of the emulator or handset

C. Monitoring thread and heap information on the Android system

D. Stopping processes

E. Simulating incoming phone calls and SMS messages to emulators

F. All of the above
2. True or False: You must use the Android emulator for debugging.

3. Which target platforms can Android applications be written for?

4. True or False: The Android emulator is a generic device that supports only one screen configuration.

**Answers**

1. F. All of the above. The DDMS perspective can be used to monitor, browse, and interact with emulators and handsets in a variety of ways.

2. False. The Android emulator is useful for debugging, but you can also connect the debugger to an actual device and debug directly.

3. There are a number of target platforms available and more are added with each new SDK release. Some important platform targets include Android 1.1, Android 1.5, Android 1.6, Android 2.0, Android 2.0.1, and Android 2.1. Targets higher than Android 1.1 can include the Google APIs, if desired. These targets map to the AVD profiles you must create in order to use the Android emulator.

4. False. The Android emulator is a generic device, but it can support several different skins. For a complete list of skins supported, see the Android SDK and AVD Manager in Eclipse.

**Exercises**

1. Launch the Android emulator and browse the settings available. Change the language settings. Uninstall an application.

2. Launch the Android emulator and customize your home screen. Change the wallpaper. Install an AppWidget. Get familiar with how the emulator tries to mimic a real handset. Note the limitations, such as how the dialer works.

3. Try launching the Hierarchy Viewer tool with the Droid1 project you created in Hour 1. Note how you can drill down to see the TextView controls you created.
A
aapt (Android Asset Packaging Tool), 39
AbsoluteLayout control, 115
accelerometer, 362
accessing
  application functionality with contexts, 47
  application preferences, 46-47
layouts, 71
LBS (location-based services)
  last known location, 245
  providers, 244
network services
  HTTP networking, 261-262
  network permissions, 260
  network status, checking, 260-261
phone status information
  retrieving telephony information, 279-280
  setting phone state permissions, 278
raw files, 72-73
strings, 64
XML files, 72
accounts, Android Market
developer accounts
  benefits, 402
  signing up for, 396-397
ACTION_CHOOSER intent, 226
ACTION_IMAGE_CAPTURE intent,
  225-226, 230
ACTION_VIEW intent, 248
activities
  activity requirements, determining, 96-97
  adding options menus to, 139
  callback methods, 49-50
  designating launch activity, 87-88
  explained, 47
  implementing for Been There, Done That! game, 105-106
  launching, 48-49
  managing state, 49
  preferences, 110
activities

registering, 86-87, 92
saving state, 50-51
activity dialogs. See dialogs
<activity> tag, 86-87
ActivityUnitTestCase class, 379
adb (Android Debug Bridge), 39, 391
addresses, translating, 247
addTab() method, 155
addTextChangedListener() method, 169
ADT (Android Development Tools)
Android Project Wizard, 10-11
installing, 412-413
overview, 9
afterTextChanged() method, 191
AIDL Compiler, 39
AlertDialog, 182
alerting users with notifications, 348-349
AndAppStore, 404
anddev.org, 425
Android Asset Packaging Tool (aapt), 39
Android Debug Bridge (adb), 39, 391
Android developer program, 8-9
Android developer website, 364, 425
Android Development Tools (ADT). See ADT
Android documentation, 27-29
Android Hierarchy Viewer, 38
Android Market, 8, 425
billing, 401
developer account benefits, 402
developer accounts benefits, 402
signing up for, 396-397
explained, 395
language support, 405
locales, 326-327
removing applications from, 402
return policy, 401-402
uploading applications to, 397-400
Android mascot, 8
Android Mobile Application Development website, 424-425
Android NDK, 23
Android Plug-in for Eclipse (ADT), installing, 412-413
Android Project command (New menu), 10
Android Project Wizard, 10-11
Android SDK
defining, 343
detecting programmatically, 343
installing, 10
explained, 411
Linux installations, 412
Mac OS X installations, 412
Windows installations, 411
setting minimum Android SDK version, 83
specifying target SDK, 342
upgrading, 413
versions, 341-342
Android Virtual Devices. See AVDs
Android Wireless Application Development, 73, 251, 364
android.bluetooth package, 363
android.database package, 359
android.database.sqlite package, 359
android.gesture package, 352
android.graphics package, 355
android.media package, 353
android.provider package, 360
android.sax.* package, 72
android.service.wallpaper package, 357
android.speech.Recognizer Intent, 353
android.speech.tts package, 352
android.util.Xml.* package, 72
AndroidManifest.xml file, 12-15
activities
designating launch activity, 87-88
registering, 86-87, 92
application permissions, managing, 88-92
application settings, configuring
linking secondary libraries, 82
naming Android packages, 82
naming applications, 84
providing application descriptions, 85
providing application icons, 84
setting debug information for applications, 85
setting minimum Android SDK version, 83
versioning applications, 82
designating launch activity in, 48
editing with Eclipse manifest file resource editor
AndroidManifest.xml tab, 81
Application tab, 78-79
Instrumentation tab, 80
Manifest tab, 78-79
Permissions tab, 79-80
explained, 77-78
<manifest> tag, 82
preparing for release, 385-386
<receiver> tag, 307
<uses-library> tag, 82
updating for App Widgets, 307
versioning applications, 83
android:debuggable attribute (<application> tag), 85
android:description attribute (<application> tag), 85
android:icon attribute (<application> tag), 84
android:label attribute (<application> tag), 84
android:minSdkVersion attribute (<uses-sdk> tag), 83
android:versionCode attribute (<manifest> tag), 82
android:versionName attribute (<manifest> tag), 82
animation
adding to splash screens, 120-121
animating all views in layout, 122-123
animating specific views, 121-122
handling animation life cycle events, 123
ImageSwitcher control, 207
performance issues, 124
types of, 119-120
annotations, @UiThreadTest, 378
App Widgets
adding to Home screen, 312-313
Android manifest file updates, 307
AppWidgetProvider class implementation, 309-311
background operations, 314-315
creating services, 316-317
starting/stoping services, 317-318
controls, 318
event handling, 313-314
explained, 305-306
layout, 308
methods, 309
multiple instances of, 319
properties, 306-307
RemoteViews object, 311
application assets
compared to project resource, 74
definition of, 73
application contexts
accessing application functionality, 47
accessing application preferences, 46-47
launching activities with, 48
retrieving application resources, 46
retrieving context for current process, 46
Application Manager, 109
application resources
definition of, 59
referencing, 62
storing, 60-62
application servers, 256-257
application sliding drawer, 37
Application tab (manifest file), 14, 78-79
<application> tag
android:debuggable attribute, 85
android:description attribute, 85
android:icon attribute, 84
android:label attribute, 84
applications
activities
callback methods, 49-50
explained, 47
launching, 48-49
managing state, 49
saving state, 50-51
How can we make this index more useful? Email us at indexes@samspublishing.com
applications

application assets
    compared to project resource, 74
    definition of, 73
application contexts
    accessing application functionality, 47
    accessing application preferences, 46-47
    retrieving application resources, 46
    retrieving context for current process, 46
application information, logging, 374
application resources
    definition of, 59
    referencing, 62
    storing, 60-62
avatars
    adding to settings screen layout, 219-220
    bitmaps, 228-230
    designing, 217-219
    gallery, 227-228
    ImageButton controls, 221-223
    launching activities and handling results, 224-225
    photo-taking with camera, 225-227
Been There, Done That! game. See Been There, Done That! application
configuring application settings
    linking secondary libraries, 82
    naming Android packages, 82
    naming applications, 84
    providing application description, 85
    providing application icons, 84
    setting debug information, 85
    setting minimum Android SDK version, 83
    version code, 82
    version name, 82
    versioning applications, 82-83
context menus, 138
copy protection, 406
designing
    application activity requirements, 44-45
    application features, 43-44
    application functionality, 45-46
    for response during low-memory conditions, 55
dialogs. See dialogs
game logic
    addressing edge cases, 213-214
    declaring string literals for question parsing, 209
handling button presses, 211-212
    storing questions in hashtable, 210
    updating
        SharedPreferences to include game state settings, 208-209
game screens
    adding resources to, 200-202
    designing, 197-200
    updating layout of, 202-203
help screens
    adding resources to, 145-146
    designing, 144
    raw resource files, 147-148
    updating layout of, 146
installing, 390-391
intents
    explained, 51
    launching other applications with, 52-53
    passing information with, 51-52
internationalizing application names, 92
launching
    in emulator, 109
    with intents, 52-53
layouts
    accessing programmatically, 71
    designing with Layout Resource Editor, 68-70
designing with XML, 69
explained, 67-68
logging application information, 54
main menu screens
adding project resources, 131
designing, 127-130
layout requirements, 129
ListView control, 129, 134-137
screen headers, building with RelativeLayout, 129
updating layouts, 132-133
naming, 84
options menus
adding resources to, 138-139
adding to activities, 139
handling menu selections, 140
permissions, 347
including in Android manifest file, 92
managing, 88-91
prototypes
activities, implementing, 105-106
activity requirements, 96-97
application preferences, creating, 106-108
creating new project, 103
debug configuration, 108
game screen features, 102
help screen features, 98-99
high-level game features, 96
main menu screen features, 98
project resources, adding, 104
scores screen features, 100
settings screen features, 100-101
splash screen features, 97-98
providing descriptions for, 85
providing icons for, 84
publishing. See publishing
removing from Android Market, 402
responsiveness, 367
scores screens
adding resources to, 151-152
completed scores screen, 157
delays in loading, 158
designing, 149
layout requirements, 150
TabHost control, 150, 155
updating layout of, 152-154
security, 367
setting debug information for, 85
settings screens
adding resources to, 165-166
Button controls, 170-172
designing, 161-163
EditText controls, 168-170
SharedPreferences,
175-178
Spinner controls, 172-174
updating layout of, 166-167
splash screens
adding resources to, 116-117
animation, 119-124
designing, 113-114
Layout controls, 114-116
updating layout of, 117-119
stability, 367
testing, 40
automated testing, 373-380
best practices, 367-370
on emulator, 372
managing test environment, 371-372
on target handsets, 373
types of testing, 370
uploading to Android Market, 397-400
verifying, 391-392
version code, setting, 82
version name, setting, 82
ViewSwitcher controls, 203-204
generating with ViewFactory, 204-205
ImageSwitcher, 206-207
TextSwitcher, 205
ApplicationTestCase class, 379

How can we make this index more useful? Email us at indexes@samspublishing.com
AppWidgetProvider class

AppWidgetProvider class, 309-311
ARCHOS 5 Internet tablet, 402
assertTrue() method, 377
assets
  compared to project resources, 74
definition of, 73
/assets folder, 12
asynchronous tasks, running
  with AsyncTask class, 265-266
  with threads and handlers, 266
AsyncTask class, 265-266, 282-285, 288, 297-298
audio, 353
authors' contact information, 425
authors' website, 424-425
Auto-complete feature, 416
automated testing, 373
  adding more tests, 379-380
  creating test cases, 375-377
  creating test projects, 374-375
  explained, 374
  logging application information, 374
  running automated tests, 378-379
availability of servers, checking, 261
avatars
  adding to settings screen layout, 219-220
  bitmaps
    generating, 229
    saving, 228-229
scaling, 229-230
transformations, 230
designing, 217-219
gallery, 227-228
ImageButton controls
  handling events, 222-223
  setting images of, 221-222
launching activities and handling results, 224-225
photo-taking with camera, 225-227
uploading, 288
AVDs (Android Virtual Devices)
  advantages, 24
  creating, 17-18, 241
background operations
  in App Widgets, 314-315
  creating services, 316-317
  starting/stopping services, 317-318
  handling, 268-269, 273
backward compatibility, designing for, 342
battery life, 363
BATTERY_STATS permission, 363
 Been There, Done That! application
  activities, 96-97, 105-106
  activity dialogs, 181
    AlertDialog class, 182
    CharacterPickerDialog class, 182
    custom password dialog, 188-193
    DatePickerDialog class, 182-187
    defining, 183
    Dialog class, 182
    dismissing, 184
    initializing, 183
    launching, 183
    life cycle of, 182-183
    ProgressDialog class, 182
    removing from use, 184
    TimePickerDialog class, 182
App Widget
  adding to Home screen, 312-313
  Android manifest file updates, 307
  AppWidgetProvider class implementation, 309-311
  background operations, 314-318
  controls, 318
  event handling, 313-314
  explained, 305-306
  layout, 308
  methods, 309
  multiple instances of, 319
  properties, 306-307
  RemoteViews object, 311
application preferences
  creating, 106-107
  retrieving shared preferences, 107-108
  saving shared preferences, 107
 avatars
   adding to settings screen layout, 219-220
   bitmaps, 228-230
   designing, 217-219
gallery, 227-228
ImageButton controls, 221-223
layout, 219-220
   launching activities and handling results, 224-225
   photo-taking with camera, 225-227
   context menus, 138
   creating new project, 103
designing, 108
   explained, 95
favorite place feature
   accessing LBS (location-based services), 244-245
designing, 233-234
dialog, 235-237
enabling location testing on emulator, 241-243
geocoding services, 246-247
guidelines for LBS (location-based services), 240-241
implementing framework for, 237-240
layout updates, 234-235
maps, 248-251
receiving location updates, 245
friend support
   displaying friends’ scores, 298
   enabling friend requests, 293-298
   enhancing player relationships, 299-300
   explained, 292-293
game logic
   addressing edge cases, 213-214
declaring string literals for question parsing, 209
   handling button presses, 211-212
   storing questions in hashtable, 210
   updating
   SharedPreferences to include game state settings, 208-209
   game screen, 102
   adding resources to, 200-202
designing, 197-200
   updating layout of, 202-203
help screen, 98-99
   adding resources to, 145-146
designing, 144
   raw resource files, 147-148
   updating layout of, 146
   high-level game features, determining, 96
   internationalization, 325-326
   launching in emulator, 109
   main menu, 98
   adding project resources, 131
designing, 127-130
   layout requirements, 129
   ListView control, 129, 134-137
   screen headers, building with RelativeLayout, 129
   updating layouts, 132-133
   network support. See network applications
   options menus
   adding resources to, 138-139
   adding to activities, 139
   handling menu selections, 140
   project resources, adding, 103-104
   scores screen, 100
   adding resources to, 151-152
   completed scores screen, 157
delays in loading, 158
designing, 149
layout requirements, 150
TabHost control, 150, 155
   updating layout of, 152-154
   settings screen, 100-101
   adding resources to, 165-166
   Button controls, 170-172
designing, 161-163
   EditText controls, 168-170
Been There, Done That! application

SharedPreferences, 175-178
Spinner controls, 172-174
updating layout of, 166-167
splash screen, 97-98
adding resources to, 116-117
animation, 119-123
designing, 113-114
Layout controls, 114-116
updating layout of, 117-119
ViewSwitcher controls, 203-204
generating with ViewFactory, 204-205
ImageSwitcher, 206-207
TextSwitcher, 205
billing (Android Market), 401
bindService() method, 317
bitmaps
generating, 229
saving, 228-229
scaling, 229-230
transformations, 230
Blog tab (Android documentation), 28
Bluetooth, 363
BroadcastReceiver class, 363
Browser application, 37
Browser content provider, 360
browsing file system with DDMS, 31
build errors, resolving, 420
Button controls
configuring, 170-171
handling button clicks, 171-172
buttons
handling button presses, 211-212
Upload Application, 397
Button_Friend_Email control, 296
call state information, retrieving, 279
callback methods (activities), 49-50
CallLog, 360
camera, 225-227
cancellation, handling, 270-271
CDMA phones, determining, 279-280
certificate authorities, 388
certification programs, 380
changing locales, 324
CharacterPickerDialog, 182
checking
network status, 260-261
server availability, 261
Chippy’s Revenge game
activity requirements, 44-45
application features, 43-44
application functionality, 45-46
choosing
target languages, 331
target locales, 331
target platform, 342
Class command (New menu), 415
classes. See specific classes
Clean command (Project menu), 420
clearAnimation() method, 122-123
clearing progress indicators, 270
clicks, handling, 171-172
client/server testing, 370
code comments, 416
code editing, 416-417
code formatting, 418
code organization, 418
coding standards, 368
colors
explained, 64-65
retrieving, 65
supported color formats, 65
commands
Class (New menu), 415
Clean (Project menu), 420
Override/Implement Methods (Source menu), 415
comments, 416
commit() method, 212
committing EditText input, 169
Community tab (Android documentation), 28
compress() method, 228
configuration management, 333-335
  debugging
    device debugging, 414
    USB debugging, 413-414
  EditText controls, 168
  power settings, 363
  ringtones, 356
  Spinner controls, 173
  TabHost control, 155
  wallpaper, 356-357
conformance testing, 370
ConnectivityManager class, 261
contacting authors, 425
Contacts application, 37
Contacts content provider, 360
contains() method, 177
containsKey() method, 210
content providers
  Browser, 360
  CallLog, 360
  Contacts, 360
  explained, 360-361
  live folders, 361
  MediaStore, 360
  UserDictionary, 360
context menus, 138
Context object
  getConfiguration() method, 329
  getSystemService() method, 261, 279
launching activities with, 48
retrieving context for current process, 46
controls. See specific controls
copy protection, 406
Create New Project in Workspace button (New Android Project dialog), 10
Create Project from Existing Sample button (New Android Project dialog), 13
Create Project from Existing Source button (New Android Project dialog), 13
create() method, 193
createChooser() method, 226
createFromResource() method, 173
createScaledBitmap() method, 229
Cube Live Wallpaper, 357
currency
  Currency class, 330
  internationalization, 330
Currency class, 330
custom dividers, adding to ListView, 136
custom log filters, 420
custom password dialog
  adding to QuizSettingsActivity class, 190-193
designing, 188-189
  implementing layout, 190
  launching, 193
custom selectors, adding to ListView, 137
custom views, 350
d() method (Log class), 54
Dalvik Debug Monitor Service.
  See DDMS
databases
  handset databases, 372
  SQLite databases, 359
DateFormat class, 330
DatePickerDialog, 182-184
  adding to QuizSettingsActivity class, 184-185
  initializing, 185-186
  launching, 186-187
date internationalization, 330
DDMS (Dalvik Debug Monitor Service)
  browsing file system with, 31
debugging Android applications with, 21-22
debugging applications with, 29-30
  Emulator Control, 31-33
eXplained, 39
File Explorer, 31
  managing tasks, 30
  managing tasks with, 30
Screen Capture button, 33-34
  simulating incoming calls to emulator, 31-32
  simulating incoming SMS messages to emulator, 33
  taking screenshots of emulator or handset, 33-34
viewing log information, 35

How can we make this index more useful? Email us at indexes@samsnzaublishing.com
debugging

Android applications with DDMS, 21-22, 29-30
debug configuration, 108
debug information, setting for applications, 85
device debugging, 414
USB debugging, 413-414
default resources, specifying, 325
default tabs, setting, 155
default.properties file, 12
defect tracking systems, 369
defects, 369
defining Android SDK, 343
deleteFile() method, 359
descriptions, adding to applications, 85
designating launch activity, 87-88
designing applications
  activity requirements, 44-45, 96-97
  App Widget layouts, 308
  application features, 43-44
  application functionality, 45-46
  avatars, 217-219
  backward compatibility, 342
  favorite place feature, 233-234
  game screens, 102, 197-200
  help screens, 98-99, 144
  high-level game features, 96
  layouts
    with Layout Resource Editor, 68-70
    with XML, 69
  main menu screens, 98
  network applications
    application servers, 256-257
    explained, 255-256
    progress bars, 257
    password dialog, 188-189
    response during low-memory conditions, 55
    scores screens, 100, 149
    settings screens, 100-101, 161-163
    splash screens, 97-98, 113-114
    input forms, 55
detecting SDK programmatically, 343
determinate progress, displaying
  with progress bars, 263
determining locales, 329-330
Dev Guide tab (Android documentation), 28
Dev Tools, 38, 110
developer accounts (Android Market)
  benefits, 402
  signing up for, 396-397
Developer Challenges, 9
Developer.com, 426
developing network applications, 257-258
device fragmentation, 371-372
designing for backward compatibility, 342
detecting SDK programmatically, 343
specifying target SDK, 342
developing for different devices
  configuration management, 333-335
  handset features, 341
  screen orientations, 335-339, 344
Dialer application, 37
Dialog class, 182
dialogs, 53-54, 181
  AlertDialog, 182
  custom password dialog
    adding to QuizSettingsActivity class, 190-193
    designing, 188-189
    implementing layout, 190
    launching, 193
  DatePickerDialog, 182
    adding to QuizSettingsActivity class, 184-185
    initializing, 185-186
    launching, 186-187
  DatePickerDialog class, 184
defining, 183
Dialog, 182
dismissing, 182
favorite place dialog, 235-237
launching, 183
life cycle of, 182-183
methods, 53
ProgressDialog, 182
removing from use, 184
TimePickerDialog, 182
digital signatures, 387-390
dimensions, 65-66
directories
/assets, 12
explained, 358
/layout, 15
live folders, 361
/res, 12, 15-16, 62
/res/drawable, 13
resource directory qualifiers, 334-335
/src, 12
/values, 16
dismiss() method, 264
dismissDialog() method, 53, 183-184
dismissing
dialogs, 184
progress dialog, 273
displaying scores, 267
background processing, 268-269
cancellation, 270-271
friends’ scores, 298
progress indicator, 268-270
progress updates, 269-270
ScoreDownloaderTask class, 267
documentation, 27-29
code comments, 416
online versus local SDK documentation, 40
doInBackground() method, 265, 268, 273, 283-285, 297
downloading
Android SDK, 9, 411
Eclipse IDE, 410
question batches, 271-273
scores
background processing, 268-269
cancellation, 270-271
progress indicator, 268-270
progress updates, 269-270
ScoreDownloaderTask class, 267
Draw 9-Patch tool, 39
drawable resources
adding to Been There, Done That! game, 104
images
loading, 67
ShapeDrawable class, 67
supported image formats, 66-67
Droid #1 project
creating, 10-11
creating debug and run configurations, 18-19
debugging with DDMS, 21-22
editing project resources
AndroidManifest.xml file, 13-15
/res files, 15-16
string resources, 16
launching on handset, 22-23
launching with emulator, 19-21
project files, 12-13

drawable resources
adding to Been There, Done That! game, 104
images
loading, 67
ShapeDrawable class, 67
supported image formats, 66-67
Droid #1 project
creating, 10-11
creating debug and run configurations, 18-19
debugging with DDMS, 21-22
editing project resources
AndroidManifest.xml file, 13-15
/res files, 15-16
string resources, 16
launching on handset, 22-23
launching with emulator, 19-21
project files, 12-13

Eclipse IDE
Auto-complete feature, 416
automated testing with
adding more tests, 379-380
creating test cases, 375-377
creating test projects, 374-375
explained, 374
running automated tests, 378-379
build errors, resolving, 420
classes, creating, 415
code comments, 416
code editing, 416-417
code formatting, 418
code organization, 418
developing Android applications without, 39
imports, organizing, 415-416
installing, 410
integrating with source control packages, 421
log filters, 420

How can we make this index more useful? Email us at indexes@samspublishing.com
manifest file resource editor
- AndroidManifest.xml tab, 81
- Application tab, 78-79
- Instrumentation tab, 80
- Manifest tab, 78-79
- Permissions tab, 79-80

methods, creating, 415
refactoring, 418-420
Rename tool, 417-418
tabs, rearranging, 421

displayed testing, 370
editing
- AndroidManifest.xml file
- AndroidManifest.xml tab, 81
- Application tab, 78-79
- Instrumentation tab, 80
- Manifest tab, 78-79
- Permissions tab, 79-80
code, 416-417
project resources
- AndroidManifest.xml file, 13-15
- /res files, 15-16
- string resources, 16
- XML files, 24

EditText controls
- committing EditText input, 169
- configuring, 168
- handling text input, 168
- listening for EditText key-strokes, 169-170

elements. See tags

emulator
- configuring location of, 241
- enabling location testing on, 241-243
- explained, 35
- incoming calls, simulating with DDMS, 31-32
- incoming SMS messages, simulating with DDMS, 33
- launching Android applications with, 19-21
- launching applications in, 109
- limitations, 35
- providing input to, 36
- SD card images with, 38
- skins, 38, 40
- taking screenshots of, 33-34
- testing network applications on, 258
- testing on, 372

Emulator Control (DDMS), 31-33

enabling
- friend requests
  - AsyncTask class, 297-298
  - Friend Request
dialog, 296
- settings screen layout,
  293-295
- location testing on emulator,
  241-243
- USB debugging, 413-414

enhancing player relationships,
  299-300

errors, build errors, 420

events
- handling animation life cycle events, 123
- handling in App Widgets, 313-314
- ImageButton events, 222-223
- ListView events
  - custom dividers, 136
  - custom selectors, 137
  - listening for, 135-136
- execute() method, 267, 298

Export Android Application command, 389

Extract Local Variable tool, 419

Extract Method tool, 419

Facebook Platform for Mobile, 301
Facebook support, 300-301

fade_in.xml animation, 120

delay_in2.xml animation, 120

favorite place feature
- accessing LBS (location-based services), 244-245
designing, 233-234
dialog, 235-237
- enabling location testing on
  emulator, 241-243
- geocoding services, 246-247
guidelines for LBS (location-based services), 240-241
- implementing framework for, 237-240
layout updates, 234-235
maps
   launching map applications with intents, 248-249
   working with Google APIs, 250-251
   receiving location updates, 245
feasibility testing, 373
FierceDeveloper, 425
File Explorer (DDMS), 31
file system, browsing with DDMS, 31
fileList() method, 359
files
   AndroidManifest.xml. See AndroidManifest.xml file
default.properties, 12
DroidActivity.java, 12
explained, 358
help.xml, 144
JAR files, 286-287
main.xml, 13
quizhelp.txt, 145
R.java class file, 12, 62
raw resource files, 72-73, 147-148
scores.xml, 150-154
strings.xml, 13
widget.xml, 308
XML files
   accessing, 72
   editing, 24
   formatting, 71
   parsing, 156-157
   retrieving, 156
   XML parsers, 74
filling ListView control, 134-135
filters, log filters, 420
findViewByld() method, 54, 147, 155, 171-173, 191
finish() method, 51
firmware, troubleshooting, 343
folders. See directories
forgoing internationalization, 327-328
format() method, 187
formatting
   code, 418
   strings, 64
   XML files, 71
forms
   Button controls
      configuring, 170-171
      handling button clicks, 171-172
   EditText controls
      committing EditText input, 169
      configuring, 168
      handling text input, 168
      listening for EditText keystrokes, 169-170
   input forms, 55
   saving form data with
   SharedPreferences
      defining
         SharedPreferences entries, 175
      reading settings from, 177-178
      saving settings to, 176
settings screens
   adding resources to, 165-166
   designing, 161-163
   updating layout of, 166-167
Spinner controls, 172
   configuring, 173
   handling Spinner selections, 173-174
   listening for selection events, 174
frame-by-frame animation, 119
FrameLayout control, 115
Friend Request dialog, 296
friend requests, enabling
   AsyncTask class, 297-298
   Friend Request dialog, 296
   settings screen layout, 293-295
friend support
   enabling friend requests
      AsyncTask class, 297-298
      Friend Request dialog, 296
      settings screen layout, 293-295
   explained, 292-293
   friends’ scores, displaying, 298
FriendRequestTask class, 296-297
fromFile() method, 228
full internationalization, 328-329
functional testing, 370
gallery, 227-228

game logic
  addressing edge cases, 213-214
  declaring string literals for question parsing, 209
  handling button presses, 211-212
  storing questions in hashtable, 210
  updating SharedPreferences to include game state settings, 208-209

game screens
  adding resources to, 200-202
  defining features of, 102
  designing, 197-200
  updating layout of, 202-203
  ViewSwitcher controls, 203-204
    generating with ViewFactory, 204-205
    ImageSwitcher, 206-207
    TextSwitcher, 205

/get/com.androidbook.droid1/R.java file, 12

generating bitmaps, 229

Geocoder class, 247

gEOcODing services, 246-247

gestures, handling, 351-352

GET method, 282-285

getActivity() method, 376
getApplicationContext() method, 46
getAssets() method, 74
getAttributeValue() method, 157

getBestProvider() method, 244
getCacheDir() method, 359
getCallState() method, 279
getColor() method, 65
getConfiguration() method, 329
getDeviceId() method, 279
getDimension() method, 66
getDir() method, 359
getDrawable() method, 67
getFilesDir() method, 359
getFromLocation() method, 247
getFromLocationName() method, 247
getIntent() method, 51
getLastKnownLocation() method, 245
getLatitude() method, 245
getLongitude() method, 245
getNetworkInfo() method, 261
getNetworkType() method, 279
getOwnerActivity() method, 194
getPackageManager() method, 392
getPhoneType() method, 279
getPreferences() method, 50, 110
getProvider() method, 244
getQuestionImageDrawable() method, 206
getQuestionImageUrl() method, 207
getResources() method, 62, 74
getSharedPreferences() method, 46, 110
getSimOperator() method, 280
getSimOperatorName() method, 280
getSimSerialNumber() method, 280
getSimState() method, 280
getString() method, 64, 177
getSubscriberId() method, 280
getSystem() method, 63
getSystemService() method, 261, 279, 362-363
getText() method, 168, 176
getVoiceMailNumber() method, 280
getWidgetData() method, 311
getXml() method, 72, 156
GIFs, animated GIFs, 119

global search, integrating with, 361

Google APIs, 241, 250-251

Google APIs Add-On Reference website, 251

Google App Engine, 256

Google Developer Challenges, 9

Google Maps, 241

Google Open Handset Alliance, 7-8

graphics
  android.graphics package, 355
  OpenGL ES graphics API, 355

GSM phones, determining, 279-280

handleAnswerAndShowNextQuestion() method, 211

chandleNoQuestions() method, 213
Handler class, 266

handsets
- handset databases, 372
- launching Android applications on, 22-23
- supporting, 341
- taking screenshots of, 33-34
- target handsets
  - identifying and acquiring, 371
  - testing on, 373

hardware
- battery life, 363
- Bluetooth, 363
- power settings, 363
- sensor data, reading, 362
- testing network applications on, 259
- Wi-Fi, 363

hash tables, storing questions in, 210

Hello, World application
- creating debug and run configurations, 18-19
- creating project, 10-11
- debugging with DDMS, 21-22
- editing project resources
  - AndroidManifest.xml file, 13-15
  - /res files, 15-16
  - string resources, 16
- launching on handset, 22-23
- launching with emulator, 19-21
- project files, 12-13

help screens
- adding resources to, 145-146
- defining features of, 98-99
- designing, 144
- raw resource files, 147-148
- updating layout of, 146

HelpActivity class, 44

high-level game features, determining, 96

history of Android, 8

Home screen
- adding App Widgets to, 312-313
- explaining, 37

HTTP GET method, 282-285

HTTP networking, 261-262

HTTP POST method, 286-288

HttpClient class, 285-286

HttpGet class, 281-282

ImageButton controls
- handling events, 222-223
- setting images of, 221-222

images
- avatars
  - adding to settings screen layout, 219-220
  - designing, 217-219
- bitmaps
  - generating, 229
  - saving, 228-229
  - scaling, 229-230
  - transformations, 230
- image media, 223
  - gallery, 227-228
  - launching activities and handling results, 224-225
  - photo-taking with camera, 225-227

ImageButton controls, 221-223

loading, 67

SD card images, 38

ShapeDrawable class, 67

supported image formats, 66-67

ImageSwitcher controls
- animating, 207
- initializing, 206
- updating, 207

ImageUploadTask class, 288

imports, organizing, 415-416

incoming calls to emulator, simulating with DDMS, 31-32

incoming SMS messages to emulator, simulating with DDMS, 33

How can we make this index more useful? Email us at indexes@samspublishing.com
indeterminate progress, displaying with progress bars, 263
inflate() method, 191
InformIT website, 423-424
integrating Eclipse IDE with source control, 421
initializing
   dialogs, 183-186
   ImageSwitcher control, 206
   TextSwitcher control, 205
input forms, designing, 55
input methods, 350
inputStreamToString() method, 147
insertScoreRow() method, 270
installing
   Android Plug-in for Eclipse (ADT), 412-413
   Android SDK, 10
      explained, 411
      Linux installations, 412
      Mac OS X installations, 412
      Windows installations, 411
   applications, 390-391
   Eclipse IDE, 410
   JDK (Java Development Kit), 410
Instrumentation tab (manifest file), 14, 80
integration testing, 370
<intent-filter> tag, 87-88
intents
   ACTION_CHOOSER, 226
   ACTION_IMAGE_CAPTURE, 225-226, 230
   ACTION_VIEW, 248
      explained, 51
   launching map applications with, 248-249
   launching other applications with, 52-53
   passing information with, 51-52
   TAKE_AVATAR_CAMERA_REQUEST, 227
   TAKE_AVATAR_GALLERY_REQUEST, 228
internationalization
   application names, 92
   currency, 330
   date/time formatting, 330
      explained, 321-322
   languages
      choosing target languages, 331
      defined, 321
   locales
      Android Market support for, 326-327
      Android SDK support for, 322-323
      changing, 324
      choosing target locales, 331
      defined, 321
      determining, 329-330
   resources, specifying
      default resources, 325
      language-specific resources, 325
      region-specific resources, 326
strategies
   forgoing internationalization, 327-328
   full internationalization, 328-329
   limited internationalization, 328
   testing, 370
   troubleshooting, 331
isNetworkRoaming() method, 280
/item> element, 138

J-K

JAR files, adding to projects, 286-287
jarsigner utility, 390
Java Development Kit (JDK), installing, 410
java.io package, 358
JavaScript Object Notation (JSON), 289
javax.xml.* package, 72
JDK (Java Development Kit), installing, 410
JSON (JavaScript Object Notation), 289
JUnit
   automated testing with
      adding more tests, 379-380
      creating test cases, 375-377
      creating test projects, 374-375
explained, 374
running automated tests, 378-379
website, 380

keystrokes, EditText, 169-170
keytool utility, 390

L

landscape mode, creating custom layout for, 335-337
languages
Android Market support for, 405
choosing target languages, 331
defined, 321
language-specific resources, specifying, 325
last known location (LBS), 245
launch activity, designating, 87-88
launching activities, 48-49
Android applications
with emulator, 19-21, 109
on handset, 22-23
with intents, 52-53, 248-249
dialogs, 183
custom password dialog, 193
DatePickerDialog to, 186-187
Layout controls, 114-116

/layout folder, 15
Layout Resource Editor, 68-70
LayoutInflater class, 71
layouts
accessing programmatically, 71
adding to Been There, Done That! game, 104
App Widgets, 308
creating custom layout for landscape mode, 335-337
designing
with Layout Resource Editor, 68-70
with XML, 69
explained, 67-68
favorite place feature, 234-235
favorite place dialog, 236-237
help screens, 146
game screens, 202-203
Layout controls, 114-116
main menu screens
adding TextView template layout, 133
layout requirements, 129
ListView control, 129-130
updating master layouts, 132-133
password dialog, 190
RelativeLayout control, 129
scores screens, 150-154
settings screens, 166-167
splash screen layouts, 117-119

LBS (Location-Based Services), 352
accessing
last known location, 245
providers, 244
enabling location testing on emulator, 241-243
favorite place feature
designing, 233-234
dialog, 235-237
implementing framework for, 237-240
layout updates, 234-235
geocoding services, 246-247
guidelines for, 240-241
maps
launching map applications with intents, 248-249
working with Google APIs, 250-251
receiving location updates, 245
libraries
graphics libraries, 355
linking, 82
life cycles of activity dialogs, 182-183
light sensor, 362
limited internationalization, 328
LinearLayout control, 115, 132
linking secondary libraries, 82
Linux
Android SDK installation, 412
device debugging configuration, 414
listen() method, 279
444

listening

listening
for EditText keystrokes, 169-170
for ListView events, 135-136
for screen orientation changes, 338, 344
for selection events, 174

ListView control, 129-130
custom dividers, 136
custom selectors, 137
filling, 134-135
listening for ListView events, 135-136

live folders, 361

loading
images, 67
scores screen, 158

loadQuestionBatch() method, 273

locales
Android Market support for, 326-327
Android SDK support for, 322-323
changing, 324
choosing target locales, 331
currency, 330
date/time formatting, 330
defined, 321
determining, 329-330
resources, specifying
default resources, 325
language-specific resources, 325
region-specific resources, 326

location testing, enabling, 241-243

Location-Based Services (LBS), 352
LocationManager class, 244
Log class, 54
log filters, 420
LogCat, 35
logic. See game logic
logs
filtering, 35
log methods, 54
logging application information, 374
viewing log information, 35

low-memory conditions, 55

M

Mac OS X
Android SDK installation, 412
device debugging configuration, 414
Eclipse IDE installation, 410

magnetic field sensor, 362

main menu screens
adding project resources, 131
defining features of, 98
designing, 127-130
layout requirements, 129
ListView control, 129-130
custom dividers, 136
custom selectors, 137
filling, 134-135
listening for ListView events, 135-136
screen headers, building with RelativeLayout, 129
updating layouts, 132-133
makeView() method, 204-205
managedQuery() method, 360
managing
activity state, 49
application permissions, 88-91
tasks with DDMS, 30

manifest file. See
AndroidManifest.xml file

Manifest tab (manifest file), 14, 78-79

<manifest> tag, 82

maps
Google Maps, 241
launching map applications with intents, 248-249
working with Google APIs, 250-251

marketplaces
AndAppStore, 404
Android Market, 8, 425
billing, 401
developer account benefits, 402
developer accounts, 396-397, 402
explained, 395
language support, 405
locales, 326-327
removing applications from, 402
return policy, 401-402
uploading applications to, 397-400
master layouts, updating for main menu screens, 132-133
MediaController control, 354
MediaPlayer class, 353
MediaRecorder class, 353
MediaStore, 360
MenuActivity class, 44
menus
context menus, 138
main menus. See main menu screens
options menus
adding resources to, 138-139
adding to activities, 139
handling menu selections, 140
MessageDigest class, 285
Messaging application, 37
methods
addTab(), 155
addTextChangedListener(), 169
afterTextChanged(), 191
assertTrue(), 377
bindService(), 317
clearAnimation(), 122-123
commit(), 212
compress(), 228
contains(), 177
containsKey(), 210
create(), 193
createChooser(), 226
createFromResource(), 173
createScaledBitmap(), 229
creating, 415
d() (Log class), 54
deleteFile(), 359
dismiss(), 264
dismissDialog(), 53, 183-184
doInBackground(), 265, 268, 273, 283-285, 297
e() (Log class), 54
execute(), 267, 298
fileList(), 359
findViewById(), 54, 147, 155, 171-173, 191
finish(), 51
format(), 187
fromFile(), 228
GET, 282-285
getActivity(), 376
getApplicationContext(), 46
getAssets(), 74
getAttributeValue(), 157
getBestProvider(), 244
cacheDir(), 359
callState(), 279
color(), 65
configuration(), 329
deviceId(), 279
dimension(), 66
dir(), 359
drawable(), 67
dir(), 359
filesDir(), 359
fromLocation(), 247
getFromLocationName(), 247
getIntent(), 51
getLastKnownLocation(), 245
getLatitude(), 245
getLongitude(), 245
getNetworkInfo(), 261
getNetworkType(), 279
getOwnerActivity(), 194
getPackageInfo(), 392
getPhoneType(), 279
getPreferences(), 50, 110
getProvider(), 244
getQuestionImageDrawable(), 206
getQuestionImageUrl(), 207
getResources(), 62, 74
getSharedPreferences(), 46, 110
getSimOperator(), 280
getSimOperatorName(), 280
getSimSerialNumber(), 280
gSimState(), 280
getString(), 64, 177
gSubscriberId(), 280
gSystem(), 63
gSystemService(), 261, 279, 362-363
gText(), 168, 176
gVoiceMailNumber(), 280
gWidgetData(), 311
gXml(), 72, 156
handleAnswerAndShowNextQuestion(), 211
handleNoQuestions(), 213
i() (Log class), 54
inflate(), 191
inputStreamToString(), 147
insertScoreRow(), 270
isNetworkRoaming(), 280
listen(), 279

How can we make this index more useful? Email us at indexes@samspublishing.com
methods

loadQuestionBatch(), 273
makeView(), 204-205
managedQuery(), 360
next(), 156
onActivityResult(), 49, 52, 224-225, 228
onAnimationEnd(), 123
onBind(), 316
onCancelled(), 270-271
OnItemClickListener(), 136
onCreate(), 49, 171, 176, 208, 268, 316
onCreateDialog(), 53, 183-185, 190, 296
onCreateOptionsMenu(), 139
onDateSet(), 185
onDeleted(), 309
onDestroy(), 49, 177, 316
onDisabled(), 309
onEnabled(), 309
onItemClick(), 135-136
OnItemSelectedListener(), 174
onPause(), 49, 122
onPostExecute(), 266, 270, 273
onPreExecute(), 265, 268, 272
onPrepareDialog(), 53, 183-186
onProgressUpdate(), 265, 269
onReceive(), 309
onResume(), 49
onRetainNonConfigurationInstance(), 338
onStartCommand(), 316
onUpdate(), 309-311
openFileInput(), 358
openFileOutput(), 359
openRawResource(), 72, 147
openStream(), 268
parse(), 248
POST, 286-288
processScores(), 269
publishProgress(), 265, 269
putExtra(), 51
putInt(), 176
putLong(), 176
putString(), 176
removeDialog(), 53, 183-184, 192-194
renaming, 417-418
requestLocationUpdates(), 245
requestRouteToHost(), 261
saveAvatar(), 225-228
setAdapter(), 173
setContentView(), 54, 268
setCurrentTabByTag(), 155
setCurrentText(), 205, 207
setEntity(), 288
setFactory(), 204
setImageBitmap(), 221
setImageDrawable(), 207
setImageResource(), 221
setImageURI(), 206, 215, 221-222
setListAdapter(), 311
setOnCheckedChangeListener(), 273
setInput(), 262, 269, 273
setIntent(), 140
setLayoutAnimation(), 122
setNegativeButton(), 192
setOnCheckedChangeListener(), 141, 171
setOnMenuItemClickListener(), 135, 141
setOnItemTouchListener(), 174
setOnKeyListener(), 169
setOnLongClickListener(), 223
setProgress(), 263
setSelection(), 141, 173
setText(), 147, 168, 205, 207
setTextViewText(), 311
setTheme(), 350
setTitle(), 192
setup(), 155, 376
showDialog(), 53, 183-186, 193-194
startActivity(), 48, 51, 123
startActivityForResult(), 49, 52, 224
startAnimation(), 122
startService(), 317
stopSelf(), 317
stopSelfResult(), 317
stopService(), 318
tearDown(), 376
updateAppWidget(), 311, 314
updateDate(), 186
v() (Log class), 54
w() (Log class), 54
minimum Android SDK version, setting, 83
mksdcard utility, 39
MobiHand, 404
monitoring battery life, 363
multimedia, 353
  audio, 353
  supported formats, 364
  video, 354
MultipartEntity class, 288
multiple App Widgets, 319
naming
  Android packages, 82
  applications, 84
network applications
  accessing network services
    HTTP networking, 261-262
    network permissions, 260
    network status, checking, 260-261
  accessing phone status information
    retrieving telephony information, 279-280
  setting phone state permissions, 278
asynchronous tasks
  with AsyncTask class, 265-266
  with threads and handlers, 266
designing
  application servers, 256-257
  explained, 255-256
  progress bars, 257
  guidelines, 257-258
  progress bars, indicating network activity with, 262
  determinate progress, 263
  indeterminate progress, 263
  progress dialogs, 263-264
  question batches, downloading and parsing, 271-272
  dismissing progress dialog, 273
  handling background processing, 273
  starting progress dialog, 272
  scores, downloading and displaying, 267
  background processing, 268-269
  cancellation, 270-271
  progress indicator, 268-270
  progress updates, 269-270
  ScoreDownloaderTask class, 267
testing
  on emulator, 258
  on hardware, 259
uploading data to servers
  determining data to send, 277-278
  explained, 281
  with HTTP GET method, 282-285
  with HTTP POST method, 286-288
network permissions, 260
network roaming information, retrieving, 280
network services, accessing
  HTTP networking, 261-262
  network permissions, 260
  network status, checking, 260-261
  network status, checking, 260-261
  network type information, retrieving, 279
NetworkInfo class, 261
New Android Project dialog
  Create New Project in Workspace button, 10
  Create Project from Existing Sample button, 13
  Create Project from Existing Source button, 13
New menu commands
  Android Project, 10
  Class, 415
next() method, 156
Notification object, 349
NotificationManager system service, 349
notifications, 348-349

How can we make this index more useful? Email us at indexes@samspublishing.com
objects

Context
  getConfiguration() method, 329
  getSystemService() method, 261, 279
Notification, 349
RemoteViews, 311
SensorManager, 362
Service
  creating, 316-317
  starting/stopping, 317-318
WifiManager, 363
onActivityResult() method, 49, 52, 224-225, 228
onAnimationEnd() method, 123
onBind() method, 316
onCancelled() method, 270-271
OnItemClickListener() method, 136
onCreate() method, 49, 171, 176, 208, 268, 316
onCreateDialog() method, 53, 183-185, 190, 296
onCreateOptionsMenu() method, 139
onDateSet() method, 185
onDeleted() method, 309
onDestroy() method, 49, 177, 316
onEnabled() method, 309
onFling gesture, 352
onItemClick() method, 135-136
OnItemTouchListener() method, 136
online Android resources, 425-426
onLongClick() method, 227
onLongPress gesture, 352
onOptionsItemSelected() method, 140
onPause() method, 49, 122
onPostExecute() method, 266, 270, 273
onPreExecute() method, 265, 268, 272
onPrepareDialog() method, 53, 183-186
onProgressUpdate() method, 265, 269
onReceive() method, 309
onResume() method, 49
onStartCommand() method, 316
onTap() method, 309
onUpdate() method, 309, 311
Open Handset Alliance, 7-8, 425
openFileInput() method, 358
openFileOutput() method, 359
OpenGL ES, 120, 355
OpenIntents, 362, 425
OpenIntents.org website, 53
openRawResource() method, 72, 147
OpenSocial initiative, 301
openStream() method, 268
operating systems
  configuring for device debugging, 414
  supported operating systems, 409
options menus
  adding resources to, 138-139
  adding to activities, 139
  handling menu selections, 140
org.w3c.dom package, 72
org.xml.sax.* package, 72
org.xmlpull.* package, 72
organizing
  code, 418
  imports, 415-416
orientation sensor, 362
Override/Implement Methods command (Source menu), 415
overriding doInBackground() method, 273

P

PackageManager class, 392
packages
  android.bluetooth, 363
  android.database, 359
  android.database.sqlite, 359
  android.gesture, 352
  android.graphics, 355
  android.media, 353
  android.provider, 360
android.service.wallpaper, 357
android.speech.Recognizer Intent, 353
android.speech.tts, 352
installing, 390-391
java.io, 358
naming, 82
packaging applications, 387-390
parse() method, 248
parsing
declaring string literals for question parsing, 209
question batches, 271-272
dismissing progress dialog, 273
handling background processing, 273
starting progress dialog, 272
SAX parser, 74
XML files, 156-157
XMLPullParser, 74
passing information with intents, 51-52
password dialog
adding to QuizSettingsActivity class, 190-193
designing, 188-189
implementing layout, 190
launching, 193
performance
of animation, 124
performance testing, 370
<permission> element, 347
permissions, 347
BATTERY_STATS, 363
including in Android manifest file, 92
managing, 88-91
network permissions, 260
phone state permissions, 278
Permissions tab (manifest file), 14, 79-80
personalization
explained, 356
ringtones, 356
wallpaper, 356-357
phone status information, accessing
retrieving telephony information
call state information, 279
CDMA/GSM information, 279-280
network roaming information, 280
network type information, 279
SIM information, 280
voice mail information, 280
setting phone state permissions, 278
PlayActivity class, 44
player relationships, enhancing, 299-300
pleaseWaitDialog control, 264
plug-ins. See ADT (Android Development Tools)
PocketGear, 404
POST method, 286-288
power settings, 363
preferences
activity preferences, 110
application preferences accessing, 46-47
creating, 106-107
debug configuration, 108
retrieving shared preferences, 107-108
saving shared preferences, 107
privacy concerns, 302
processScores() method, 269
programming languages, support for, 23
progress bars, 257
clearing, 270
indicating network activity with, 262
determinate progress, 263
indeterminate progress, 263
progress dialogs, 263-264
starting, 268
progress dialogs, 263-264
dismissing, 273
starting, 272
progress updates, handling, 269-270
ProgressDialog class, 182, 263-264
ProGuard, 406
Project menu commands, Clean, 420

How can we make this index more useful? Email us at indexes@samspublishing.com
projects

adding to Eclipse, 13
creating, 10-11, 103
debug and run configurations, 18-19
debugging with DDMS, 21-22, 29-30
developing without Eclipse, 39
editing project resources
AndroidManifest.xml file, 13-15
/res files, 15-16
string resources, 16
launching
on handset, 22-23
with emulator, 19-21
project files, 12-13
project resources, adding, 103-104

properties of App Widgets, 306-307

prototypes
activities, 105-106
application preferences
creating, 106-107
retrieving shared preferences, 107-108
saving shared preferences, 107
debug configuration, 108
designing
activity requirements, 96-97
creating new project, 103
game screen features, 102

providers. See content providers
ProviderTestCase2 class, 380
proximity sensor, 362
publishing
on Android Market
billing, 401
developer account benefits, 402
explained, 395
removing applications, 402
return policy, 401-402
signing up for developer account, 396-397
uploading applications, 397-400
release process, 383-385
packaging and signing, 387-390
preparing release candidate build, 385-386
testing packaged application, 390-392
testing release candidate, 386-387
self-distribution, 402-403
to other marketplaces, 404-405
publishProgress() method, 265, 269
putExtra() method, 51
putInt() method, 176
putString() method, 176

question batches, downloading and parsing, 271-272
dismissing progress dialog, 273
handling background processing, 273
starting progress dialog, 272

<question> element, 201
questions
declaring string literals for question parsing, 209
storing in hashtable, 210

<questions> element, 201
QuizActivity class, 96
QuizGameActivity class, 97, 277
QuizHelpActivity class, 97
QuizMenuActivity class, 97
QuizScoresActivity class, 97
QuizSettingsActivity class, 97, 277
adding DatePickerDialog to, 184-185
adding password dialog to, 190-193
onCreateDialog() method, 296
QuizSplashActivity class, 96
QuizTask class, 271-272
QuizWidgetProvider class, 316

R
R.java class file, 62
raw resource files, 72-73
   accessing, 147-148
   adding, 147
raw sensor data, reading, 362
reading
   raw sensor data, 362
   settings from
      SharedPreferences, 177-178
ranging tabs, 421
<receiver> element, 307
receiving location updates, 245
refactoring, 418-420
Reference tab (Android documentation), 28
referencing
   application resources, 62
   system resources, 63
region-specific resources, specifying, 326

registering
   activities, 86-87, 92
   for Android Market developer accounts, 396-397
regular versioned builds, 368-369
relationships, enhancing, 299-300
RelativeLayout control, 115, 129, 132
release builds, 383
release candidate
defined, 383
packaging and signing, 387-390
preparing, 385-386
testing, 386-387
release process, 383-385
packaging and signing, 387-390
preparing release candidate build, 385-386
testing packaged application, 390-392
testing release candidate, 386-387
RemoteViews object, 311
removeDialog() method, 53, 183-184, 192-194
removing
   applications from Android Market, 402
dialogs, 184
Rename tool, 417-418
renaming items, 417-418
representational state transfer (REST), 300
requestLocationUpdates() method, 245
requestRouteToHost() method, 261
/res folder, 12, 15-16, 62
/res/drawable, 13
/res/layout/help.xml, 144
/res/layout/main.xml, 13
/res/layout/scores.xml, 150
/res/raw/quizhelp.txt, 145
/res/values/strings.xml, 13
resource directory qualifiers, 334-335
resources
adding
to game screens, 200-202
to help screens, 145-146
to main menu screens, 131
to options menus, 138-139
to scores screens, 151-152
to settings screens, 165-166
to splash screens, 116-117
Android Mobile Application Development website, 424-425
application resources, 46
definition of, 59
referencing, 62
storing, 60-62
colors
explained, 64-65
retrieving, 65
supported color formats, 65

How can we make this index more useful? Email us at indexes@samspublishing.com
resources

compared to project assets, 74
dimensions explained, 65-66
 supported dimension units, 66
drawable resources, 104
ingredients editing
 AndroidManifest.xml file, 13-15
 /res files, 15-16
 string resources, 16
explained, 73
files. See files images
 loading, 67
 ShapeDrawable class, 67
 supported image formats, 66-67
InformIT website, 423-424
layouts
 accessing programmatically, 71
 adding to Been There, Done That! game, 104
designing with Layout Resource Editor, 68-70
designing with XML, 69
explained, 67-68
online Android resources, 425-426
raw resource files, 72-73, 147-148

specifying
 default resources, 325
 language-specific resources, 325
 region-specific resources, 326
storing, 74
strings
 accessing, 64
 adding to Been There, Done That! game, 104
explained, 64
formatting, 64
system resources
definition of, 59
documentation for, 74
referencing, 63
storing, 63
XML resources,
retrieving, 156
responsive applications, 367
REST (representational state transfer), 300
results, launching activities for, 48
retrieving
color resources, 65
shared preferences, 107-108
XML files, 156
return policy (Android Market), 401-402
reverse-geocoding, 247
RingtoneManager class, 356
ringtones, 356

running
automated tests, 378-379
tasks asynchronously
with AsyncTask class, 265-266
with threads and handlers, 266

S
sandboxes, 386
saveAvatar() method, 225-228
saving
activity state, 50-51
bitmaps, 228-229
settings to
 SharedPreferences, 176
shared preferences, 107
SAX parser, 74
scaling bitmaps, 229-230
<score> element, 152
ScoreDownloaderTask class, 267
scores. See also scores screens
 downloading and displaying, 267
 background processing, 268-269
cancellation, 270-271
progress indicator, 268-270
progress updates, 269-270
ScoreDownloaderTask class, 267
uploading, 285
scores screens
  adding resources to, 151-152
completed scores
  screen, 157
  defining features of, 100
delays in loading, 158
designing, 149
layout requirements, 150
TabHost control
  adding tabs to, 155
  adding to scores
  screen, 150
  configuring, 155
  setting default tab, 155
  updating layout of, 152-154
scores.xml file, 152-154
ScoresActivity class, 44
Screen Capture button
  (DDMS), 33-34
screen headers, building with
  RelativeLayout control, 129
screen orientations
  creating custom layout for
  landscape mode, 335-337
  handling, 339
  listening for screen orientation changes, 338, 344
screenshots, taking of emulator
  or handset, 33-34
SD card images, 38
SDK (Android), 28
  defining, 343
  detecting
    programmatically, 343
installing
  explained, 411
  Linux installations, 412
  Mac OS X
    installations, 412
  Windows
    installations, 411
  specifying target SDK, 342
  upgrading, 413
  versions, 341-342
searching, global search, 361
secure applications, 367
Secure Hash Algorithm (SHA), 285
security
  application permissions, 347
  copy protection, 406
self-distribution, 402-403
sensor data, reading, 362
SensorManager object, 362
servers
  application servers, 256-257
  availability, checking, 261
  uploading data to
    determining data to send, 277-278
    explained, 281
    with HTTP GET method, 282-285
    with HTTP POST method, 286-288
service objects
  creating, 316-317
  starting/stopping, 317-318
services
  creating, 316-317
  NotificationManager system
  service, 349
  starting/stopping, 317-318
ServiceTestCase class, 380
setAdapter() method, 173
setContentView() method, 54, 268
setCurrentTabByTag() method, 155
setCurrentText() method, 205-207
setEntity() method, 288
setFactory() method, 204
setImageBitmap() method, 221
setImageDrawable() method, 206-207, 221
setImageResource() method, 221
setImageURI() method, 206, 215, 221-222
setImageViewBitmap() method, 311
setImageViewResource() method, 311
setInAnimation() method, 207
setInput() method, 262, 269, 273
setIntent() method, 140
setLayoutAnimation() method, 122
setNegativeButton() method, 192
setOnClickListener() method, 141, 171
setOnClickPendingIntent() method, 314
setOnItemClickListener() method, 135, 141

How can we make this index more useful? Email us at indexes@samspublishing.com
setOnItemSelectedListener() method, 174
setOnKeyListener() method, 169
setOnLongClickListener() method, 223
setProgress() method, 263
setSelection() method, 141, 173
setText() method, 147, 168, 205-207
setTextViewText() method, 311
setTheme() method, 350
Settings application, 38
settings screens
adding avatars to, 219-220
adding resources to, 165-166
Button controls
configuring, 170-171
handling button clicks, 171-172
defining features of, 100-101
designing, 161-163
EditText controls
committing EditText input, 169-170
configuring, 168
handling text input, 168
SharedPreferences
defining
SharedPreferences entries, 175
reading settings from, 177-178
saving, 107
saving settings to, 176
updating to include game state settings, 208-209
setTitle() method, 192
setup() method, 155, 376
SHA (Secure Hash Algorithm), 285
ShapeDrawable class, 67
SharedPreferences, 46-47
defining SharedPreferences entries, 175
reading settings from, 177-178
retrieving, 107-108
saving, 107
saving settings to, 176
updating to enable friend requests, 293-295
<form data-with Shared Preferences
defining
SharedPreferences entries, 175
reading settings from, 177-178
saving settings to, 176
overview, 358
SQLite databases, 359
SHOP4APPS, 404
showDialog() method, 53, 183-186, 193-194
signing applications, 387-390
SIM information, retrieving, 280
simulating
incoming calls to emulator, 31-32
incoming SMS messages to emulator, 33
skins (emulator), 38-40
SlideME, 404
sliding drawer, 37
SMS messages, simulating
incoming SMS messages to emulator, 33
SmsManager class, 281
SmsMessage class, 281
social features
friend support
displaying friends’ scores, 298
enabling friend requests, 293-298
enhancing player relationships, 299-300
explained, 292-293
privacy concerns, 302
social networking services, integrating Android applications with explained, 300
Facebook support, 300-301
OpenSocial initiative, 301
other social network applications, 303
Twitter support, 301
supporting player relationships, 292
tailoring to application, 292
types of, 291
social networking services, integrating Android applications with explained, 300
Facebook support, 300-301
OpenSocial initiative, 301
other social network applications, 303
Twitter support, 301
social trivia game. See Been There, Done That! application
software piracy, protecting against, 406
source control, integrating with Eclipse, 421
source files, editing, 416-417
Source menu commands, Override/Implement Methods, 415
space requirements, 410
speech
converting text to, 352
converting to text, 353
speech recognition, 353
Spinner controls, 172
configuring, 173
handling Spinner selections, 173-174
listening for selection events, 174
splash screens
adding resources to, 116-117
animation
adding to splash screens, 120-121
animating all views in layout, 122-123
animating specific views, 121-122
handling animation life cycle events, 123
performance issues, 124
types of, 119-120
defining features of, 97-98
designing, 113-114
Layout controls, 114-116
updating layout of, 117-119
SplashActivity class, 44
SQLite databases, 359
/src folder, 12
stable applications, 367
Stack Overflow: Android website, 426
startActivity() method, 48, 51, 123
startActivityForResult() method, 49, 52, 224
startAnimation() method, 122
starting
progress dialog, 272
progress indicators, 268
services, 317-318
startService() method, 317
state, activity state
callback methods, 49-50
managing, 49
saving, 50-51
status
network status, checking, 260-261
phone status information, accessing
retrieving telephony information, 279-280
setting phone state permissions, 278
stopping services, 317-318
stopSelf() method, 317
stopSelfResult() method, 317
stopService() method, 318
storing data
application resources, 60-62
directories, 358
files, 358
overview, 358
questions in hashtable, 210
resources, 74
SQLite databases, 359
system resources, 63
strategies for internationalization

forgoing internationalization, 327-328
full internationalization, 328-329
limited internationalization, 328

strings
accessing, 64
adding to Been There, Done That! game, 104
adding to settings screens, 165-166
declaring string literals for question parsing, 209
editing, 16
explained, 64
formatting, 64

styles, 349-350

supplementary materials
Android Mobile Application Development website, 424-425
InformIT website, 423-424
online Android resources, 425-426

supported operating systems, 409

system resources
definition of, 59
documentation for, 74
referencing, 63
storing, 63

TabHost control
adding tabs to, 155
adding to scores screen, 150
configuring, 155
setting default tab, 155

TableLayout control, 115
tabs. See TabHost control
tags
<activity>, 86-87
<application>
   android:debuggable attribute, 85
   android:description attribute, 85
   android:icon attribute, 84
   android:label attribute, 84
<intent-filter>, 87-88
=item>, 138
<manifest>, 82
<permission>, 347
<question>, 201
<questions>, 201
<receiver>, 307
<score>, 152
<uses-library>, 82
<uses-permission>, 88
<uses-sdk>, 83

TAKE_AVATAR_CAMERA_REQUEST intent, 227
TAKE_AVATAR_GALLERY_REQUEST intent, 228
target handsets
identifying and acquiring, 371
testing on, 373
target platforms, choosing, 342
target SDK, specifying, 342
tasks
asynchronous tasks
running with AsyncTask class, 265-266
running with threads and handlers, 266
managing with DDMS, 30
tearDown() method, 376
telephony information, retrieving
call state information, 279
CDMA/GSM information, 279-280
network roaming information, 280
network type information, 279
SIM information, 280
voice mail information, 280
TelephonyManager class
getCallState() method, 279
gerDeviceld() method, 279
gerNetworkType() method, 279
gerPhoneType() method, 279
gerSimOperator() method, 280
gerSimOperatorName() method, 280
gerSimSerialNumber() method, 280
gerSimState() method, 280
gerSubscriberId() method, 280
gerVoiceMailNumber() method, 280
isNetworkRoaming() method, 280
temperature sensor, 362
Test Application Project Wizard, 375
testing
  applications, 40
  automated testing, 373
    adding more tests, 379-380
    creating test cases, 375-377
    creating test projects, 374-375
    explained, 374
    logging application information, 374
    running automated tests, 378-379
  best practices, 367-368
    coding standards, 368
    defect tracking system, 369
    regular versioned builds, 368-369
    test plans, 369-370
  on emulator, 372
  feasibility testing, 373
  managing test environment
    device fragmentation, 371-372
    handset databases, 372
    target handsets, 371
  network applications
    on emulator, 258
    on hardware, 259
  packaged applications, 390-392
  release candidate, 386-387
  on target handsets, 373
test cases, creating, 375-377
  test environment, managing device fragmentation, 371-372
    handset databases, 372
    target handsets, 371
test plans, 369-370
test projects, creating, 374-375
types of testing, 370
text
  converting speech to, 353
  text input, handling in EditText controls, 168
  TTS (text to speech), 352
TextSwitcher controls, 205
TextView control, 133
themes, 349-350
Thread class, 266, 315
time internationalization, 330
TimePickerDialog, 182
TimeUtils class, 330
title attribute (&lt;item&gt; element), 138
tools. See specific tools /tools folder, 27
transformations, 230
translating addresses/coordinates, 247
trivia game. See Been There, Done That! application
troubleshooting
  firmware upgrades, 343
  internationalization, 331
  TTS (text to speech), 352
tweened animation, 119
Twitter support, 301

@UiThreadTest annotation, 378
UI threads, 50
Update Threads button, 30
updateAppWidget() method, 311, 314
updateDate() method, 186
updating
  game screen layout, 202-203
  help screen layout, 146
  ImageSwitcher control, 207
  scores screen layout, 152-154
  settings screen layout, 166-167
  SharedPreferences to include game state settings, 208-209
  splash screen layout, 117-119
  TextSwitcher control, 205
upgrades, testing, 370
upgrading Android SDK, 413
Upload Application button, 397
uploading
  applications to Android Market, 397-400
  avatars, 288
  player scores, 285
to servers
  determining data to send, 277-278
  explained, 281
  with HTTP GET method, 282-285
  with HTTP POST method, 286-288
UrlEncodedFormEntity class

UrlEncodedFormEntity class, 298
usability testing, 370
USB debugging, enabling, 413-414
user gestures, handling, 351-352
user interfaces
  custom views, 350
  input methods, 350
  speech recognition, 353
  styles, 349-350
  themes, 349-350
  TTS (text to speech), 352
  user gestures, handling, 351-352
UserDictionary content provider, 360
users
  alerting with notifications, 348-349
  identifiers, 302
  informing about network activity, 257
<uses-library> tag, 82
<uses-permission> tag, 88
<uses-sdk> tag, 83

V

v() method (Log class), 54
/values folder, 16
variables, renaming, 417-418
verifying applications, 391-392
versioning applications, 82-83
video, 354
Videos tab (Android documentation), 28

W

w() method (Log class), 54
wallpaper, 356-357
WallpaperManager class, 356
websites
  anddev.org, 425
  Android developer website, 364, 425
  Android Market, 425
  Android Mobile Application Development website, 424-425
  Developer.com, 426
  FierceDeveloper, 425
  InformIT, 423-424
JUnit, 380
Open Handset Alliance, 425
OpenIntents, 425
Stack Overflow: Android, 426
Wireless Developer Network, 426
Wi-Fi, 363
widget.xml file, 308
widgets. See App Widgets
WifiManager object, 363
Windows
  Android SDK installation, 411
device debugging configuration, 414
Eclipse IDE installation, 410
Wireless Developer Network, 426
wizards, Test Application Project Wizard, 375

X-Y-Z

XML files
  accessing, 72
designing layouts with, 69
editing, 24
formatting, 71
parsing, 156-157
retrieving, 156
XML parsers, 74
XML utility packages, 71-72
XmlPullParser class, 74, 262, 269
XmlResourceParser class, 156-157
zipalign utility, 390