LEARNING Android APPLICATION PROGRAMMING FOR THE Kindle Fire

A Hands-On Guide to Building Your First Android Application

LAUREN DARCEY
SHANE CONDER
Praise for Learning Android Application Programming for the Kindle Fire

“Now is a great time to learn how to program for the Kindle Fire, and this book is the perfect companion for your journey! Distilled within the text are the proven techniques, best practices, and hard-won wisdom of two of the mobile industry’s leading pioneers. You’ll be programming Kindle Fire apps in no time!”
—Mark Hammonds, Mobile Engineer, and Managing Editor, Mobiletuts+

“Learning Android Application Programming for the Kindle Fire is a must-have developers’ resource specific to the Kindle Fire. This book takes you from SDK installation to APK publication with lots of examples and tips along the way.”
—Jim Hathaway, Web Developer

“If you want to bring your application to Amazon’s exciting new Android-based tablet, look no further than this book. Darcey and Conder show you how to go from idea to application for the Kindle Fire quickly and clearly with a sample application that shows you everything you’ll need to build your application. Along the way, you’ll pick up valuable tips on developing software in general.”
—Ray Rischpater, Senior Research Engineer, Nokia Research Center

“I used Lauren Darcey and Shane Conder’s books to start developing Android apps fast. I knew iOS development but needed to learn the basics of Android development. This was a great book to show me how, and I started coding the same day.”
—Scott Walker, Developer
Visit informit.com/learningseries for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you’ve learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.
Learning Android™ Application Programming for the Kindle Fire™
This page intentionally left blank
Learning Android™ Application Programming for the Kindle Fire™

A Hands-On Guide to Building Your First Android Application

Lauren Darcey
Shane Conder

Addison-Wesley
Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Android™ is a trademark of Google, Inc. Kindle Fire™ is a trademark of Amazon.com, Inc.

Android™ and Google® are trademarks of Google, Inc. Kindle Fire™ and Amazon.com® are trademarks of Amazon.com, Inc. Neither Amazon.com, Inc., nor Google, Inc., have authorized or approved publication of this work and references to their marks herein are not intended to imply their sponsorship or affiliation with this work.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the web: informit.com/aw

The Library of Congress cataloging-in-publication data is on file.

Copyright © 2012 Lauren Darcey and Shane Conder

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.


Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing: July 2012
For Ellie
# Contents

Introduction  1

Part I: Kindle Fire Fundamentals

1 Getting Started with Kindle Fire  9
2 Mastering the Android Development Tools  29
3 Building Kindle Fire Applications  39
4 Managing Application Resources  53
5 Configuring the Android Manifest File  69
6 Designing an Application Framework  83

Part II: Building an Application Framework

7 Implementing an Animated Splash Screen  101
8 Implementing the Main Menu Screen  117
9 Developing the Help and Scores Screens  133
10 Collecting User Input  149
11 Using Dialogs to Collect User Input  165
12 Adding Application Logic  181
13 Adding Network Support  199
14 Exploring the Amazon Web Services SDK for Android  225

Part III: Publishing Your Kindle Fire Application

15 Managing Alternative and Localized Resources  233
16 Testing Kindle Fire Applications  249

17 Registering as an Amazon Application Developer  261

18 Publishing Applications on the Amazon Appstore  271

Part IV: Appendixes

A Configuring Your Android Development Environment  279

B Eclipse IDE Tips and Tricks  289

C Supplementary Materials  299
# Table of Contents

## Introduction 1

### I: Kindle Fire Fundamentals

1. **Getting Started with Kindle Fire** 9  
   - Introducing Android 9  
     - Google and the Open Handset Alliance 9  
     - Android Makes Its Entrance 10  
     - Cheap and Easy Development 10  
   - Familiarizing Yourself with Eclipse 12  
     - Creating Android Projects 13  
     - Exploring Your Android Project Files 16  
     - Editing Project Resources 17  
   - Running and Debugging Applications 20  
     - Managing Android Virtual Devices 20  
     - Creating Debug and Run Configurations in Eclipse 22  
     - Launching Android Applications Using the Emulator 22  
     - Debugging Android Applications Using DDMS 25  
     - Launching Android Applications on a Device 25  
   - Summary 28  
   - Exercises 28

2. **Mastering the Android Development Tools** 29  
   - Using the Android Documentation 29  
   - Debugging Applications with DDMS 31  
     - Managing Tasks 32  
     - Browsing the Android File System 33  
     - Taking Screenshots of the Emulator or Device 34  
     - Viewing Log Information 35  
   - Working with the Android Emulator 36  
     - Providing Input to the Emulator 36  
   - Using Other Android Tools 36  
   - Summary 36
3 Building Kindle Fire Applications  39
  Designing an Android Application  39
    Designing Application Features  40
    Determining Application Activity Requirements  40
    Implementing Application Functionality  41
  Using the Application Context  42
    Retrieving Application Resources  42
    Accessing Application Preferences  42
    Accessing Other Application Functionality Using Contexts  43
  Working with Activities  43
    Launching Activities  43
    Managing Activity State  44
    Shutting Down Activities  45
  Working with Intents  46
    Passing Information with Intents  47
    Using Intents to Launch Other Applications  47
  Working with Dialogs  48
  Working with Fragments  49
  Logging Application Information  50
  Summary  51
  Exercises  51

4 Managing Application Resources  53
  Using Application and System Resources  53
    Working with Application Resources  53
    Working with System Resources  56
  Working with Simple Resource Values  57
    Working with Strings  57
    Working with Colors  57
    Working with Dimensions  58
  Working with Drawable Resources  59
    Working with Images  59
    Working with Other Types of Drawables  60
  Working with Layouts  60
    Designing Layouts Using the Layout Resource Editor  61
    Designing Layouts Using XML  62
5 Configuring the Android Manifest File 69
Exploring the Android Manifest File 69
    Using the Manifest Tab 70
    Using the Application Tab 71
    Using the Permissions Tab 71
    Using the Instrumentation Tab 72
    Using the AndroidManifest.xml Tab 73
Configuring Basic Application Settings 73
    Naming Android Packages 74
    Versioning an Application 74
    Setting the Minimum Android API Version 75
    Naming an Application 76
    Providing an Icon for an Application 76
    Providing an Application Description 76
    Setting Debug Information for an Application 77
    Setting Other Application Attributes 77
Defining Activities 77
    Registering Activities 77
    Designating the Launch Activity 78
Managing Application Permissions 79
Managing Other Application Settings 81
Summary 81
Exercises 81

6 Designing an Application Framework 83
Designing an Android Trivia Game 83
    Determining High-Level Game Features 83
    Determining Activity Requirements 84
    Determining Screen-Specific Game Features 85
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>II: Building an Application Framework</strong></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Implementing an Animated Splash Screen</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Designing the Splash Screen</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Implementing the Splash Screen Layout</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>Adding New Project Resources</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>Updating the Splash Screen Layout</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>Working with Animation</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>Adding Animation Resources</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>Animating Specific Views</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>Setting the Image Animations</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td>Handling Animation Lifecycle Events</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>116</td>
</tr>
<tr>
<td>8</td>
<td>Implementing the Main Menu Screen</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Designing the Main Menu Screen</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Determining Main Menu Screen Layout Requirements</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Designing the Screen Header</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Designing the GridView Control</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Finishing Touches for the Main Menu Layout Design</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>Implementing the Main Menu Screen Layout</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>Adding New Project Resources</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Updating the Main Menu Screen Layout Files</td>
<td>121</td>
</tr>
</tbody>
</table>
Working with the GridView Control  124
  Filling a GridView Control  124
  Listening for GridView Events  127
Working with Other Menu Types  128
  Adding an Options Menu to the Game Screen  129
Summary  131
Exercises  131

9 Developing the Help and Scores Screens  133
  Designing the Help Screen  133
  Implementing the Help Screen Layout  135
    Adding New Project Resources  135
    Updating the Help Screen Layout  135
  Working with Files  136
    Adding Raw Resource Files  136
    Accessing Raw File Resources  137
  Designing the Scores Screen  138
    Determining Scores Screen Layout Requirements  138
    Adding the TabHost Control  139
  Implementing the Scores Screen Layout  141
    Adding New Project Resources  141
    Updating the Scores Screen Layout  142
  Building a Screen with Tabs  144
    Configuring the TabHost Control  144
    Adding Tabs to the TabHost Control  145
    Setting the Default Tab  145
  Working with XML  146
    Retrieving XML Resources  146
    Parsing XML Files with XmlResourceParser  146
    Applying Finishing Touches to the Scores Screen  147
Summary  148
Exercises  148

10 Collecting User Input  149
  Designing the Settings Screen  149
  Implementing the Settings Screen Layout  151
    Adding New Project Resources  151
    Updating the Settings Screen Layout  154
Using Common Form Controls  155
  Working with EditText Controls  156
  Working with Spinner Controls  159
Saving Form Data with SharedPreferences  161
  Defining SharedPreferences Entries  161
  Saving Settings to SharedPreferences  161
  Reading Settings from SharedPreferences  162
Summary  163

11 Using Dialogs to Collect User Input  165
  Working with Activity Dialogs  165
    Exploring the Different Types of Dialogs  166
    Tracing the Lifecycle of a Dialog  167
  Using the DatePickerDialog Class  168
    Adding a DatePickerDialog to a Class  168
    Initializing a DatePickerDialog  170
    Launching DatePickerDialog  170
  Working with Custom Dialogs  171
    Adding a Custom Dialog to the Settings Screen  172
Summary  178
Exercises  178

12 Adding Application Logic  181
  Designing the Game Screen  181
  Implementing the Game Screen Layout  183
    Adding New Project Resources  184
    Updating the Game Screen Layout  185
  Working with ViewSwitcher Controls  186
    Initializing Switcher Controls  187
    Implementing Switcher Factory Classes  187
    Updating the TextSwitcher Control  189
    Updating the ImageSwitcher Control  189
  Wiring Up Game Logic  190
    Adding Game State Settings to the SharedPreferences  191
    Retrieving, Parsing, and Storing Book Data  192
Summary  197
Exercises  197
13 Adding Network Support 199
Designing Network Applications 199
Working with an Application Server 199
Managing Lengthy Network Operations 200
Informing the User of Network Activity 201
Developing Network Applications 201
Enabling Network Testing on the Emulator 202
Testing Network Applications on Kindle Fire 202
Accessing Network Services 202
Planning Have You Read That? Network Support 202
Setting Network Permissions 203
Checking Network Status 203
Using HTTP Networking 203
Indicating Network Activity with Progress Bars 204
Displaying Indeterminate Progress 204
Displaying Determinate Progress 204
Displaying Progress Dialogs 204
Running Tasks Asynchronously 206
Using AsyncTask 206
Using Threads and Handlers 207
Downloading and Displaying Score Data 207
Extending AsyncTask for Score Downloads 207
Starting the Progress Indicator with onPreExecute() 208
Clearing the Progress Indicator with onPostExecute() 209
Handling Cancellation with onCancelled() 209
Handling Processing with doInBackground() 210
Handling Progress Updates with onProgressUpdate() 211
Starting the ScoreDownloaderTask 212
Downloading and Parsing Batches of Books 213
Extending AsyncTask for Book Downloads 213
Starting the Progress Dialog with onPreExecute() 214
Dismissing the Progress Dialog with onPostExecute() 214
Handling the Background Processing 215
Starting the BookListDownloaderTask 215
Determining What Data to Send to the Server 216
Keeping Player Data in Sync 216
Uploading Settings Data to a Remote Server  217
Working with Android Services  218
Implementing UploadTask  220
Uploading Player Data with the HTTP GET Method  220
Uploading Score Data to a Remote Server  223
Downloading Friends’ Score Data  224
Summary  224
Exercises  224

14 Exploring the Amazon Web Services SDK for Android  225
The 10,000-Foot View of AWS  225
Exploring the AWS Offerings  226
Using AWS Database and Storage Services  227
Using AWS Messaging and Notification Services  227
Using AWS Infrastructure and Administrative Services  228
Summary  228
Exercises  228

III: Publishing Your Kindle Fire Application

15 Managing Alternative and Localized Resources  233
Using the Alternative Resource Hierarchy  233
Understanding How Resources Are Resolved  234
Organizing Alternative Resources with Qualifiers  234
Using Alternative Resources Programmatically  236
Organizing Application Resources Efficiently  236
Customizing the Application Experience  237
Updating the Main Screen  237
Updating the Splash Screen  238
Updating the Game Screen  239
Updating the Other Screens  240
Internationalizing Android Applications  240
How Android Localization Works  241
How the Android Operating System Handles Locale  241
How Applications Handle Locales  241
How Kindle Fire Handles Locales  243
Contents

Android Internationalization Strategies    243
   Forgoing Application Internationalization  244
   Limiting Application Internationalization  245
   Implementing Full Application Internationalization  245
Using Localization Utilities    246
   Determining System Locale    246
   Formatting Strings Like Dates and Times    246
   Handling Currencies    247
Summary    247
Exercises    247

16 Testing Kindle Fire Applications    249
Testing Best Practices    249
   Developing Coding Standards    250
   Performing Regular Versioned Builds    250
   Using a Defect Tracking System    251
   Developing Good Test Plans    251
Maximizing Test Coverage    252
   Testing on the Emulator    252
   Testing on Target Devices    253
   Performing Automated Testing    253
Summary    260
Exercises    260

17 Registering as an Amazon Application Developer    261
Understanding the Release Process    261
Preparing the Release Candidate Build    263
   Preparing the Android Manifest File for Release    263
   Protecting Your Application from Software Pirates    264
   Readying Related Services for Release    265
Testing the Application Release Candidate    265
Signing Up as an Amazon App Developer    266
Packaging and Signing an Application    267
   Digitally Signing Applications    267
   Exporting and Signing the Package File    267
Contents

C Supplementary Materials 299
Using the Source Code for This Book 299
Accessing the Android Developer Website 300
Accessing the Publisher’s Website 300
Accessing the Authors’ Website 301
Contacting the Authors 303
Leveraging Online Android Resources 304

Index 305
Acknowledgments

This book would never have been written without the guidance and encouragement we received from a number of very patient and supportive people, including our editorial team, coworkers, friends, and family.

Throughout this project, our editorial team at Pearson has been top notch. Special thanks go to Laura Lewin and Trina MacDonald, Olivia Basegio, Sheri Cain, and the rest of the Pearson publishing team. Our technical reviewers, Ray Rischpater, Mike Wallace, and Jim Hathaway, helped us ensure that this book provides accurate information. With each edition, this book gets better. However, it wouldn’t be here without the help of many folks on past editions. Thanks go out to past reviewers, technical editors, and readers for their valuable feedback. Finally, we want to thank our friends and family members who supported us when we needed to make our deadlines.
About the Authors

Lauren Darcey is responsible for the technical leadership and direction of a small software company specializing in mobile technologies, including Android, Apple iOS, BlackBerry, Palm Pre, BREW, and J2ME and consulting services. With more than two decades of experience in professional software production, Lauren is a recognized authority in application architecture and the development of commercial-grade mobile applications. Lauren received a B.S. in Computer Science from the University of California, Santa Cruz.

Lauren spends her free time traveling the world with her geeky mobile-minded husband and daughter. She is an avid nature photographer whose work has been published in books and newspapers around the world. In South Africa, she dove with 4-meter-long great white sharks and got stuck between a herd of rampaging hippopotami and an irritated bull elephant. She’s been attacked by monkeys in Japan, gotten stuck in a ravine with two hungry lions in Kenya, gotten thirsty in Egypt, narrowly avoided a coup d’état in Thailand, geocached her way through the Swiss Alps, drank her way through the beer halls of Germany, slept in the crumbling castles of Europe, and gotten her tongue stuck to an iceberg in Iceland (while being watched by a herd of suspicious wild reindeer).

Shane Conder has extensive development experience and has focused his attention on mobile and embedded development for the past decade. He has designed and developed many commercial applications for Android, Apple iOS, BREW, BlackBerry, J2ME, Palm, and Windows Mobile—some of which have been installed on millions of phones worldwide. Shane has written extensively about the mobile industry and evaluated mobile-development platforms on his tech blogs and is well known within the blogosphere. Shane received a B.S. in Computer Science from the University of California.

A self-admitted gadget freak, Shane always has the latest smartphone, tablet, or other mobile device. He can often be found fiddling with the latest technologies, such as cloud services and mobile platforms, and other exciting, state-of-the-art technologies that activate the creative part of his brain. He is a very hands-on geek dad. He also enjoys traveling the world with his geeky wife, even if she did make him dive with 4-meter-long great white sharks and almost get eaten by a lion in Kenya. He admits that he has to take at least two phones with him when backpacking—even though there is no coverage—and that he snickered and whipped out his Android phone to take a picture when Laurie got her tongue stuck to that iceberg in Iceland, and that he has learned that he should be writing his own bio.

The authors have also published several other Android books, including Android Wireless Application Development, Android Wireless Application Development Volume I: Android Essentials, Android Wireless Application Development Volume 2: Advanced Topics, Sams Teach Yourself Android Application Development, and the mini-book Introducing Android Development with Ice Cream Sandwich. Lauren and Shane have also published numerous articles on mobile-software development for magazines, technical journals, and online publishers of educational content. You can find dozens of samples of their work in Linux User and Developer, Smart Developer magazine (Linux New Media), developer.com, Network World, Envato (MobileTuts+ and
CodeCanyon), and InformIT, among others. They also publish articles of interest to their readers at their own Android website: http://androidbook.blogspot.com. You can find a full list of the authors’ publications at http://goo.gl/f0Vlj.

We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn’t like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book. Please contact the authors for technical help related to this book at androidwirelessdev+kf1@gmail.com.

When you write, please be sure to include this book’s title and author as well as your name and email address. We will carefully review your comments and share them with the authors and editors who worked on the book.

Email: errata@informit.com

Mail: Addison-Wesley
ATTN: Reader Feedback
1330 Avenue of the Americas
35th Floor
New York, New York, 10019

Reader Services

Visit our website and register this book at http://informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
This page intentionally left blank
Introduction

The Amazon Kindle Fire is one of the most exciting and popular new Android devices available to consumers and application developers alike. It’s so popular, in fact, that in the six weeks the Amazon Kindle Fire was available for purchase at the end of 2011, 3.9 million units were sold, accounting for over 14 percent of the tablet market, making it the #1-selling Android tablet (http://goo.gl/f9hFM). This book covers Android Fundamentals for Kindle Fire application development. Here, you are introduced to Android, become familiar with the Android SDK and tools, install the development tools, write your first Android application, and deploy it to a Kindle Fire device. You are also introduced to the design principles necessary to write Android applications, including how Android applications are designed, developed, and published on the Amazon Appstore.

Target Audience for This Book

There’s no reason anyone with an Amazon Kindle Fire device, a good idea for a mobile application, and some programming knowledge 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 can help you realize your goals of making killer Android apps for devices like the Kindle Fire.

We make as few assumptions about you, as a reader of this book, as possible. No wireless development experience is necessary. 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). We also assume that you own at least one Amazon Kindle Fire device and can navigate your way around it for testing purposes.

Android apps are written in Java. Therefore, we assume you have a reasonably solid understanding of the Java programming language (classes, methods, scoping, object-oriented programming [OOP], and so on), ideally using the Eclipse development environment. Familiarity with common Java packages, such as java.lang, java.net, and java.util, will serve you well.

Android can also be a fantastic platform for learning Java, provided that you have some background in OOP and adequate support, such as a professor or some really good Java programming references. We make every attempt to avoid using any fancy or confusing Java in this book, but you will find that, with Android, certain syntactical Java wizardry not often covered
in your typical beginner’s Java book is used frequently: anonymous inner classes, method chaining, templates, reflection, and so on. With patience—and some good Java references—even beginning Java developers should be able to make it through this book alive; those with a solid understanding of Java should be able to take this book and run with it without issue.

Finally, regardless of your specific skill set, we expect you to use this book in conjunction with other supplementary resources, specifically the Android SDK reference and the sample source code that accompanies each coding chapter. The Android SDK reference provides exhaustive documentation about each package, class, and method of the Android SDK. It’s searchable online. If we were to duplicate this data in book form, this book would weigh a ton—literally.

How This Book Is Structured

In just a handful of straightforward lessons, you’ll design and develop a fully functional Android application specifically for Amazon Kindle Fire devices. Each lesson builds on your knowledge of newly introduced Android concepts, and you’ll iteratively improve your application from chapter to chapter. You’ll also find numerous exercises at the end of each chapter to help cement your newfound knowledge.

This book is divided into four parts:

- **Part I: Kindle Fire Fundamentals**— Here, you 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. You also learn how to start developing for a specific Android device: the Amazon Kindle Fire.

- **Part II: Building an Application Framework**— In this part, you begin developing an application called Have You Read That?, which serves as the primary teaching tool for the rest of this book. You start by developing its animated splash screen, followed by screens for the main menu, settings, help, and scores. You learn basic user-interface design principles, how to collect input from the user, and how to display dialogs to the user. You implement the core game logic and leverage the network. Finally, you explore some of the secondary issues of application development, like internationalization.

- **Part III: Publishing Your Kindle Fire Application**— Here, you learn what you need to do to prepare for and publish your Android applications to the Amazon Appstore.

- **Part IV: Appendixes**— Here is where you can find several helpful references for setting up your Android development environment for Amazon Kindle Fire development, using the Eclipse IDE, and accessing supplementary book materials, such as this book’s websites and downloadable source code.
What Is (and Isn’t) in This Book

First and foremost, this book provides a thorough introduction to the Android platform for Kindle Fire application development. In this, we begin with the fundamentals, try to cover the most important aspects of development, and provide information on where to go for more information. This is not an exhaustive reference on the Android SDK. We assume that you will use this book as a companion to the Android SDK documentation, available for download as part of the SDK and online at http://developer.android.com.

We only have a short amount of time to get you, the reader, up to speed on the fundamentals of Android development, so forgive us if we stay strictly to the topic at hand. Therefore, we take the prerequisites listed earlier seriously. This book will not teach you how to program, explain Java syntax and programming techniques, or stray too far into the details of supporting technologies often used by mobile applications, like algorithm design, network protocols, developing web servers, graphic design, database schema design, and other such peripheral topics; there are fantastic references available on each of these subjects.

The Android SDK and related tools are updated frequently (every few months). This means that, no matter how hard we try, some minor changes in step-by-step instructions may occur if you choose to use versions of the tools that do not match those listed in the next section, “Development Environment Used.” This book is written for Kindle Fire app developers, so it focuses on the Android SDK version used by this specific device.

Because this book helps developers write applications for the Amazon Kindle Fire, we specifically targeted Android SDK Version 2.3.4 (API Level 10) for the tutorial in this book. Still, we make every effort to make this book compatible with other versions of Android, as well as work smoothly regardless of what version of the Android SDK you want to target.


Development Environment Used

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

- Windows 7 and Mac OS X 10.7.x.
- Eclipse Java IDE Version 3.7 (Indigo).
- Android ADT Plug-in for Eclipse, 16.0.1.
Introduction

- Android SDK Tools, Release 16.
- Sun Java SE Development Kit (JDK) 6 Update 21.
- Code examples target Android SDK API Level 10.
- The code was tested on the original Android Kindle Fire (Android SDK 2.3.4, API Level 10).
- The network portions of the sample application leverage Google App Engine.

Conventions Used in This Book

This book uses the following code-related conventions:

- Code and programming terms are set in a monospace font.
- ➥ signifies 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 this 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.”
About the Short Links

We chose to make many Internet links in this book short links. This benefits the readers of the print book by making it far easier and less prone to error when typing links. These links are all shortened with the goo.gl link shortener, which is a service provided by Google. If the target of the link goes away, neither the original link nor the shortened link would work. We’re confident that this is the easiest way for readers to effectively use the links we provide. In addition, as authors, we get to see which links readers are actually using.

Sometimes, link shorteners are used as a way to hide nefarious links. Be assured that we have only shortened links that we believe to be valid and safe. In addition, Google provides screening of the target URLs for malware, phishing, and spam sites. Should a target link change hands and become a bad link, using the shortened link will provide you, the reader, with an extra layer of protection.

For more information on this subject, see http://www.google.com/support/websearch/bin/answer.py?answer=190768 (http://goo.gl/iv8c7).

Code Examples for This Book

The source code is available for download on the publisher’s website (www.informit.com/title/9780321833976) and on the authors’ website (http://androidbook.blogspot.com/p/book-code-downloads.html).

We provide complete, functional code projects for each coding chapter in this book. If you’re having trouble building the tutorial application as you go along, compare your work to the sample code for that chapter. The sample code is not intended to be the “answer,” but it is the complete code listings that could not otherwise be reproduced in a book of this length.

Supplementary Tools Available

Shane Conder and Lauren Darcey, the authors, also run a blog at http://androidbook.blogspot.com, where you can always download the latest source code for their books. This website also covers a variety of Android topics and reader discussions, questions, clarifications, the occasional exercise walkthrough, and lots of other information about Android development. You can also find links to their various technical articles online and in print.

Contacting the Authors

Feel free to contact us if you have specific questions; we often post addendum information or tool-change information on our book website:

http://androidbook.blogspot.com

You can also email us at androidwirelessdev+kf1@gmail.com.
This page intentionally left blank
Getting Started with Kindle Fire

Android is the first complete, open, and free mobile platform. Developers enjoy a comprehensive software development kit (SDK), with ample tools for developing powerful, feature-rich applications. The platform is open source, relying on tried-and-true open standards with which developers will be familiar. Best of all, there are no costly barriers to entry for developers: no required fees. (A modest fee is required to publish on third-party distribution mechanisms, such as the Android Market.) Android developers have numerous options for distributing and commercializing their applications.

Introducing Android

To understand where Android fits with other mobile technologies, let’s take a minute to talk about how and why this platform came about.

Google and the Open Handset Alliance

In 2007, a group of handset manufacturers, wireless carriers, and software developers (notably, Google) formed the Open Handset Alliance, with the goal of developing the next generation of wireless platform. Unlike existing platforms, this new platform would be nonproprietary and based on open standards, which would lead to lower development costs and increased profits. Mobile software developers would also have unprecedented access to the handset features, allowing for greater innovation.

As proprietary platforms, such as RIM BlackBerry and Apple iPhone, gained traction, the mobile development community eagerly listened for news of this potential game-changing platform.
Android Makes Its Entrance

In 2007, the Open Handset Alliance announced the Android platform and launched a beta program for developers. Android went through the typical revisions of a new platform. Several preview versions of the Android SDK were released. The first Android handset (the T-Mobile G1) began shipping in late 2008. Throughout 2009 and 2010, new and exciting Android smartphones reached markets throughout the world, and the platform proved itself to industry and consumers alike. Over the last three years, numerous revisions to the Android platform have been rolled out, each providing compelling features for developers to leverage and users to enjoy. Recently, mobile platforms began to consider devices above and beyond the traditional smartphone paradigm to other devices, such as tablets, ebook readers, and set-top boxes, like Google TV.

As of this writing, hundreds of Android devices are available to consumers around the world—from high-end smartphones to low-end “free with contract” handsets and everything in between. This figure does not include the numerous Android tablet and e-book readers also available, the dozens of upcoming devices already announced, or the consumer electronics running Android. (For a nice list of Android devices, check out this Wikipedia link: http://goo.gl/fU2X5.) More than 450,000 applications are currently published on the Android Market (now called Google Play), and there are more than 30,000 applications on the Amazon Appstore for Android. In the United States, all major carriers now carry Android phones prominently in their product lines, as do many in Asia, Europe, Central/South America, and beyond. The rate of new Android devices reaching the world markets continues to increase.

Google has been a contributing member of the Open Handset Alliance from the beginning. The company hosts the Android open source project and the developer website (http://developer.android.com). This website is your go-to site for downloading the Android SDK, getting the latest platform documentation, and browsing the Android developer forums. Google also runs the most popular service for selling Android applications to end users: the Android Market. The Android mascot is a little green robot (see Figure 1.1).

Cheap and Easy Development

If there's one time when “cheap and easy” is a benefit, it's with mobile development. Wireless application development, with its ridiculously expensive compilers and preferential developer programs, has been notoriously expensive to break into compared to desktop development. Here, Android breaks the proprietary mold. Unlike other mobile platforms, there are virtually no costs to developing Android applications.

The Android SDK and tools are freely available on the Android developer website (http://developer.android.com [http://goo.gl/K8GgD]). The freely available Eclipse program has become the most popular integrated development environment (IDE) for Android application development; there is a powerful plug-in available on the Android developer site for facilitating Android development with Eclipse.
Introducing Android

So, we covered cheap; now let’s talk about why Android development is easy. Android applications are written in Java, which is one of the most popular development languages around. Java developers will be familiar with many of the packages provided as part of the Android SDK, such as java.net. Experienced Java developers will be pleased to find that the learning curve for Android is reasonable.

In this book, we focus on the most common, popular, and simple setup for developing Android applications:

- We use the most common and supported development language: Java. Although we do not teach you Java, we try our best to keep the Java code we use simple and straightforward so that even beginners won’t wrestle with syntax. Even so, if you are new to Java, we recommend Sams Teach Yourself Java in 24 Hours by Rogers Cadenhead and Thinking in Java by Bruce Eckel, Fourth Edition in Print (Third Edition free from http://goo.gl/mtj0z—a zip file from Bruce Eckel’s site at http://www.mindviewinc.com/Books/).

- We use the most popular development environment: Eclipse. It’s free, it’s well supported by the Android team, and it’s the only supported IDE compatible with the Android Development Tools plug-in. Did we mention it’s free?
We write instructions for the most common operating system used by developers: Windows. Users of Linux or Mac may need to translate some keyboard commands, paths, and installation procedures.

- We focus on the Android platform version available on the Amazon Kindle Fire: Android 2.3.4 (API Level 10).

If you haven't installed the development tools needed to develop Android applications or the Android SDK and tools yet, do so at this time.

Let's get started!

Note

You can find all the details of how to install and configure your computer for Android application development in Appendix A, “Configuring Your Android Development Environment.” You will need to install and configure Java, Eclipse, the Android SDK, and the ADT plug-in for Eclipse. You will need to configure ADB and possibly USB drivers for connecting your development machine to Kindle Fire for debugging. Again, all this is covered in Appendix A.

Familiarizing Yourself with Eclipse

Let's begin by writing a simple Android “Hello, World” application that displays a line of text to the user. As you do so, you also tour the Eclipse environment. Specifically, you learn about some of the features offered by the Android development tools (ADT) plug-in for Eclipse. The ADT plug-in provides functionality for developing, compiling, packaging, and deploying Android applications. Specifically, the ADT plug-in provides the following features:

- The Android project wizard, which generates all the required project files
- Android-specific resource editors, including a graphical layout editor for designing Android application user interfaces
- The Android SDK and the AVD Manager
- The Eclipse DDMS perspective for monitoring and debugging Android applications
- Integration with the Android LogCat logging utility
- Integration with the Android Hierarchy Viewer layout utility
- Automated builds and application deployment to Android emulators and devices
- Application packaging and code signing tools for release deployment, including ProGuard support for code optimization and obfuscation

Now, let’s take some of these features for a spin.
Creating Android Projects

The Android project wizard creates all the required files for an Android application. Open Eclipse and follow these steps to create a new project:

1. Choose File, New, Android Project or click the Android Project creator icon on the Eclipse toolbar.

   **Note**
   The first time you try to create an Android Project in Eclipse, you might need to choose File, New, Project, and then select Android, Android Project. After you do this once, the Android project type appears in the Eclipse project types, and you can use the method described in step 1.

2. Choose a project name. In this case, name the project **HelloKindle**.

3. Choose a location for the project source code. Because this is a new project, select the Create New Project in Workspace radio button. If you prefer to store your project files in a location other than the default, simply uncheck the Use Default Location checkbox and browse to the directory of your choice. The settings should look like Figure 1.2.

![Figure 1.2  Project Name and Location](image)
4. Click the Next button.

5. Select a build target for your application, as shown in Figure 1.3. For most applications, you want to select the version of Android that’s most appropriate for the devices used by your target audience and the needs of your application. For Kindle development, choose API Level 10 (Android 2.3.3) using the Android Open Source Project vendor version (not the Google, Inc., vendor version). Kindle Fire devices do not have access to Google add-ons.

![Figure 1.3 Choose SDK Target](image)

6. Click the Next button.

7. Specify an application name. This name is what users will see. In this case, call the application Hello Kindle.
8. Specify a package name, following standard package-namespace conventions for Java. Because all code in this book falls under the com.kindlebook.* namespace, use the package name com.kindlebook.hellokindle.

9. If needed, check the Create Activity checkbox. This instructs the wizard to create a default launch activity class for the application. Call your activity HelloKindleActivity.

10. Confirm that the Minimum SDK field is correct. This field will be set to the API level of the build target by default. (Android 2.3.3 is API Level 10.) If you want to support older versions of the Android SDK, you need to change this value. For example, to support devices with Android 1.6, set the Minimum SDK field to API Level 4. The Kindle is based on API Level 10, however, so an application just targeting the Kindle does not need to worry about this. Your project settings will look like what’s shown in Figure 1.4.

11. The Android project wizard allows you to create a test project in conjunction with your Android application, also shown in Figure 1.4. For this example, a test project is unnecessary. However, you can always add a test project later by clicking the Android Test Project creator icon, which is to the right of the Android project wizard icon on the Eclipse toolbar. Test projects are discussed further in Chapter 16, “Testing Kindle Fire Applications.”

12. Click the Finish button.
You can also add existing Android projects to Eclipse by using the Android project wizard. To do this, simply select Create Project from Existing Source instead of the default Create New Project in Workspace in the New Android Project dialog (refer to Figure 1.2). Several sample projects are provided in the /samples directory of the Android SDK, under the specific platform they support. For example, the Android SDK sample projects are found in the directory /platforms/android-xxx/samples (where xxx is the platform level number, such as 10).

You can also select a third option: Create Project from Existing Sample, which will do as it says. However, make sure that you choose the build target first option to get the list of sample projects that you can choose.

Note

Exploring Your Android Project Files

You will now see a new Android project called HelloKindle in the Eclipse File Explorer. In addition to linking the appropriate Android SDK jar file, the following core files and directories are created:

- AndroidManifest.xml—The central configuration file for the application.
- project.properties—A generated build file used by Eclipse and the Android ADT plug-in. Do not edit this file.
- proguard.cfg—A generated build file used by Eclipse, ProGuard, and the Android ADT plug-in. Edit this file to configure your code optimization and obfuscation settings for release builds.
- /src folder—Required folder for all source code.
- /src/com.kindlebook.hellokindle/HelloKindleActivity.java—Main entry point to this application, named HelloKindleActivity. This activity has been defined as the default launch activity in the Android manifest file.
- /assets folder—Required folder where uncompiled file resources can be included in the project.
- /res folder—Required folder where all application resources are managed. Application resources include animations, drawable graphics, layout files, data-like strings and numbers, and raw files.
- /res/drawable- folders—Application icon graphic resources are included in several sizes for different device screen resolutions.
- /res/layout/main.xml—Layout resource file used by HelloKindleActivity to organize controls on the main application screen.
- /res/values/strings.xml—The resource file where string resources are defined.
**Editing Project Resources**

The Android manifest file is the central configuration file for an Android application. Double-click the AndroidManifest.xml file within your new project to launch the Android manifest file editor (see Figure 1.5).

![Figure 1.5 Editing the Android Manifest File in Eclipse](image)

**Editing the Android Manifest File**

The Android manifest file editor organizes the manifest information into numerous tabs:

- **Manifest**—This tab, shown in Figure 1.5, is used for general application-wide settings, such as the package name and application version information (used for installation and upgrade purposes).
Chapter 1  Getting Started with Kindle Fire

- **Application**—This tab is used to define application details, such as the name and icon the application displays, as well as the “guts” of the application, such as what activities can be run (including the default launch `HelloKindleActivity`) and other functionality and services that the application provides.

- **Permissions**—This tab is used to define the application’s permissions. For example, if the application requires the ability to access Internet resources, it must register a `uses-permission` tag within the manifest, with the name `android.permission.INTERNET`.

- **Instrumentation**—This tab is used for unit testing, using the various instrumentation classes available within the Android SDK.

- **AndroidManifest.xml**—This tab provides a simple XML editor to directly edit the manifest file. Because all Android resource files, including the Android manifest file, are simply XML files, you can always edit the XML instead of using the resource editors. You can create a new Android XML resource file by clicking the Android XML creator icon on the Eclipse toolbar.

If you switch to the AndroidManifest.xml tab, your manifest file will look something like this:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.kindlebook.hellokindle"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="10" />
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
<activity
    android:name=".HelloKindleActivity"
    android:label="@string/app_name">
<intent-filter>
    <action
        android:name="android.intent.action.MAIN" />
    <category
        android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

### Editing Other Resource Files

Android applications are made up of functions (Java code, classes) and data (including resources like graphics, strings, and so on). Most Android application resources are stored under the `/res`
subdirectory of the project. The following subdirectories are also available by default in a new Android project:

- **/drawable-ldpi, /drawable-hdpi, /drawable-mdpi**—These subdirectories store graphics and drawable resource files for different screen densities and resolutions. If you browse through these directories using the Eclipse Project Explorer, you will find the `icon.png` graphics file in each one; this is your application’s icon.

- **/layout**—This subdirectory stores user interface layout files. Within this subdirectory, you will find the `main.xml` screen layout resource file, which defines the user interface for the one activity in this simple application.

- **/values**—This subdirectory organizes the various types of resources, such as text strings, color values, and other primitive types. Here, you find the `strings.xml` resource file, which contains all the string resources used by the application.

If you double-click any of the resource files, the resource editor launches. Remember that you can always directly edit the XML. For example, let’s try editing a string resource file. If you inspect the `main.xml` layout file of the project, you notice that it displays a simple layout with a single `TextView` control. This user-interface control simply displays a string. In this case, the string displayed is defined in the string resource called `@string/hello`. To edit the string resource called `@string/hello` using the string resource editor, follow these steps:

1. Open the `strings.xml` file in the resource editor by double-clicking it in the Package Explorer of Eclipse.
2. Select the String called `hello` and note the name (`hello`) and value (`Hello World, HelloKindleActivity!`) shown in the resource editor.
3. Within the Value field, change the text to `Hello, Kindle Fire`.
4. Save the file.

If you switch to the `strings.xml` tab and look through the raw XML, you notice that two string elements are defined within a `<resources>` block:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello, Kindle Fire</string>
  <string name="app_name">Hello Kindle</string>
</resources>
```

The first resource is the string called `@string/hello`. The second resource is the string called `@string/app_name`, which contains the name label for the application. If you look at the Android manifest file again, you see `@string/app_name` used in the application configuration.

We talk more about project resources in Chapter 4, “Managing Application Resources.” For now, let’s move on to compiling and running the application.
Running and Debugging Applications

To build and debug an Android application, you must first configure your project for debugging. The ADT plug-in enables you to do this entirely within the Eclipse development environment. Specifically, you need to do the following:

- Create and configure an Android Virtual Device (AVD).
- Create an Eclipse debug configuration for your project.
- Build the Android project and launch the emulator with the new AVD.

When you complete each of these tasks, Eclipse attaches its debugger to the Android emulator (or Android device connected via USB), and you are free to run and debug the application as desired.

Managing Android Virtual Devices

To run an application in the Android emulator, you must configure an AVD. The AVD profile describes the type of device you want the emulator to simulate, including which Android platform to support. You can specify different screen sizes and resolutions, and you can specify whether the emulator has an SD card and, if so, its capacity. In this case, a slightly modified AVD for the default installation of Android 2.3.3 will suffice. Here are the steps for creating a basic AVD:

1. Launch the Android Virtual Device Manager from within Eclipse by clicking the little Android icon with the bugdroid in mini-phone on the toolbar. You can also launch the manager by selecting Window, AVD Manager in Eclipse.
2. Click the New button to create a new AVD.
3. Choose a name for the AVD. Because you are going to take all the defaults, name this AVD KindleFire-Portrait.
4. Choose a build target. The Kindle Fire is based on API Level 10 (Android 2.3.3). Remember not to use a Google API version, because Kindle Fire does not support this.
5. Choose an SD card capacity, in either kibibytes or mibibytes. (Not familiar with kibibytes? See this Wikipedia entry: http://goo.gl/N3Rdd.)

**Note**

Although to mimic a Kindle Fire, you’d choose 8GiB, we recommend choosing something fairly small, because a file of the size of the SD card will be allocated on your drive each time you create a new AVD; these can add up quickly. Unless your application requires substantial storage, we recommend something like 64MiB.

6. Choose a skin. This option controls the different visual looks of the emulator. In this case, we use the effective resolution of the Kindle Fire screen of 600 pixels wide and
1004 pixels high (the default portrait resolution). Alternatively, we could create an AVD for landscape mode, where we’d need to use 1024-pixels wide and 580-pixels high. The Kindle Fire reserves some space for a soft key menu.

7. Under Hardware, change the Abstracted LCD Density to 169 and change the Device RAM Size to 512 to better emulate the Kindle Fire device characteristics.

8. Optionally, enable the Snapshot feature. This allows you to save and restore the state of an emulator session, which dramatically improves the speed with which it launches.

Your project settings should look like what’s shown in Figure 1.6.

9. Click the Create AVD button and wait for the operation to complete. This may take a few seconds if your SD card capacity is large, because the memory allocated for the SD card emulation is formatted as part of the AVD creation process. You should now see your newly created AVD in the list.

![Create new Android Virtual Device (AVD)](image)

Figure 1.6  Creating a New AVD in Eclipse
Creating Debug and Run Configurations in Eclipse

You are almost ready to launch your application. You have one last task remaining: You need to create a debug configuration (or a run configuration) for your project in Eclipse. To do this, follow these steps:

1. In Eclipse, choose Run, Debug Configurations from the menu or, alternatively, click the dropdown menu next to the debug icon on the Eclipse toolbar and choose the Debug Configurations option.

2. Double-click the Android Application item to create a new entry.

3. Edit that new entry, currently called New_configuration, by clicking it in the left pane.

4. Change the name of the configuration to HelloKindleDebug.

5. Set the project by clicking the Browse button and choosing the HelloKindle project.

6. On the Target tab, check the box next to the AVD you created.

7. Apply your changes by clicking the Apply button. Your Debug Configurations dialog should look like what's shown in Figure 1.7.

![Debug Configurations Dialog](image)

Figure 1.7 The HelloKindleDebug Debug Configuration in Eclipse

Launching Android Applications Using the Emulator

It's launch time, and your application is ready to go! To launch the application, you can simply click the Debug button from within the Launch Configuration screen, or you can do
it from the project by clicking the little green bug icon \(\text{ inoc }\) on the Eclipse toolbar. Then, select HelloKindleDebug debug configuration from the list.

**Note**

The first time you try to select HelloKindleDebug debug configuration from the little green bug dropdown, you have to navigate through the Debug Configuration Manager. Future attempts will show the HelloKindleDebug configuration for convenient access.

After you click the Debug button, the emulator launches (see Figure 1.8). This can take some time, so be patient.

![Image](Android Emulator Home Screen)
Now, the Eclipse debugger is attached, and your application runs, as shown in Figure 1.9.

As you can see, the application is simple. It displays a single `TextView` control with a line of text. The application does nothing else.

The emulator’s home screen doesn’t look anything like the home screen on a real Kindle Fire device, because it has been redesigned by Amazon. Among other things, this means that the
emulator won’t work for full application testing. You need to get a real Kindle Fire device for that.

Controlling the Emulator
When you create a custom AVD in this way, it will not have the keyboard and control buttons to the left of the screen, like you might be used to with the default emulators. All the commands are available through your development machine keyboard. For example, the Home key maps conveniently to the Home key. The menu key maps to F2 or page-up. Search maps to F5. Back maps to Esc. There are many more; find them in the Android documentation at http://goo.gl/5DMiI.

Debugging Android Applications Using DDMS
In addition to the normal Debug perspective built into Eclipse for stepping through code and debugging, the ADT plug-in adds the DDMS perspective. While you have the application running, quickly look at this perspective in Eclipse. You can get to the DDMS perspective (see Figure 1.10) by clicking the Android DDMS icon in the top-right corner of Eclipse. To switch back to the Eclipse Project Explorer, simply choose the Java perspective from the top-right corner of Eclipse.

The DDMS perspective can be used to monitor application processes, as well as interact with the emulator. You can simulate voice calls and send SMS messages to the emulator. You can send a mock location fix to the emulator to mimic location-based services. You learn more about DDMS and the other tools available to Android developers in Chapter 2, “Mastering the Android Development Tools.”

The LogCat logging tool is displayed on both the DDMS perspective and the Debug perspective. This tool displays logging information from the emulator or the device, if a device is plugged in via USB.

Launching Android Applications on a Device
It’s time to load your application onto a real Kindle Fire device. To do this, you need to connect the Kindle Fire to your computer using a USB data cable. Make sure that you have your machine configured for Kindle Fire debugging, as discussed in Appendix A.

To ensure that you debug using the correct settings, follow these steps:

1. In Eclipse, from the Java perspective (as opposed to the DDMS perspective), choose Run, Debug Configurations.
3. On the Target tab, change Deployment Target Selection Mode to Manual. You can always change it back to Automatic later, but choosing Manual forces you to choose whether to debug within the emulator (with a specific AVD) or a device, if one is plugged in via USB, whenever you choose to deploy and debug your application from Eclipse.
Apply your changes by clicking the Apply button.

Plug a Kindle Fire device into your development computer by using a USB cable.

Click the Debug button within Eclipse. A dialog appears (see Figure 1.11), showing all available configurations for running and debugging your application. All physical devices are listed, as are existing emulators that are running. You can also launch new emulator instances by using other AVDs that you have created.

Choose the available Kindle Fire device. If you do not see the Kindle Fire listed, check your cables and make sure that you installed the appropriate drivers, as explained in Appendix A.

Eclipse now installs the Android application onto your Kindle Fire, attaches the debugger, and runs your application. Your device will show a screen similar to the one you saw in the emulator. If you look at the DDMS perspective in Eclipse, you see that logging information is available, and many features of the DDMS perspective work with physical devices and the emulator, such as taking a screenshot (see Figure 1.12).
Figure 1.11   Choosing an Application Deployment Target

Figure 1.12   The Application Running on a Kindle Fire
New to Eclipse?
If you’re still learning the ropes of the Eclipse development environment, now is a great time to check out Appendix B, “Eclipse IDE Tips and Tricks.”

Summary
Congratulations! You are now a Kindle Fire Android developer. You have begun to learn your way around the Eclipse development environment. You created your first Android project. You reviewed and compiled working Android code. Finally, you ran your newly created Android application on the Android emulator and on a real Kindle Fire.

Exercises
2. Visit the Eclipse website and look around. Check out the online documentation at http://www.eclipse.org/documentation/ (http://goo.gl/fc406). Eclipse is an open source project made freely available; check out the Contribute link (http://www.eclipse.org/contribute/) and consider how you might give back to this great project in some way—either by reporting bugs or doing one of the many other options suggested.
Index

accessing
Amazon Web Services SDK, 226
Android Developer websites, 300
applications
functionality, 44
preferences, 42-43
author's website, 301-303
communication protocols, 199
networks
resources, 80
services, 202-204
permissions, 79-81, 203
publisher's websites, 300
raw files, 64
raw resource files, 137
resources, 53
XML, 64
accounts
developers
benefits of, 273-274
registering, 271
registering, 266
activities
applications, implementing, 92-93
applying, 43-45
defining, 77-79
launching, 43-44, 78-79
networks, 201
options menus, adding, 130
ProgressBar control, 204-205
registering, 77-78
shutting down, 45
state, managing, 44-45
trivia games, 84-85
Activity class, 15, 40-41
DatePickerDialog class, 168-171
dialogs, 165-168
customizing, 171-177
types of, 166-167
layout files, 40
methods, 48
adding
DatePickerDialog to classes, 168-169
dialogs, customizing, 172-175
drawable resources, 92
existing projects, 15
GridView control, 122-123
headers, scores screens, 147
locales, 241
logic to applications, 181
network support, 199
options menus, 129-130
resources, 91-92, 135
colors, 153
main menu screens, 120-121
raw files, 136
splash screens, 103-106
strings, 153
tabbing support, 139
TabHost control, 139
tabs, 145
testing, 259
XML files, 58
addTab() method, 145
ADTs (Android Development Tools), 12, 280
installing, 282
advanced graphics animation, 110
alarms, 44
AlertDialog, 166, 173
alternative hierarchies, resources, 233-236
alternative resources. See also resources
programming, 236
qualifiers, managing, 234-236
rules, 234
Amazon
Appstore, 244
Developer Portal, 304
developers, registering as, 266
publishing, 271. See also publishing
reports, 274
Amazon

royalties, 273
selling on, 271-274
CloudWatch, 228
DynamoDB, 227
Elastic Load Balancing, 228
S3, 227
Simple Email Service (SES), 227
Simple Notification Service (SNS), 227
Simple Queue Service (SQS), 228
Web Services SDK, 225
navigating, 226-228
overview of, 225-226

<application> tag, 76
Application Nodes section, 78
applications
activities, implementing, 92-93
attributes, 77
builds, 20-28, 39
activities, 43-45
context, 42-43
dialog boxes, 48-49
fragments, 49-50
intents, 46-48
logging, 50-51
tracking, 74
customizing, 237-240
DDMS, debugging, 31-35
deployment, 12
descriptions, 76
design, 39-41
development, 11
devices, launching, 25-28
DRM, 265
environments, configuring, 279-280
frameworks
design, 83
trivia games, 83-89
functionality
accessing, 44
implementing, 41
Have You Read That?, 84. See also Have You Read That? application
icons, 76
internationalizing, 240-241, 245-246
JDK, installing, 281
launching, 22-25, 47
locales, 241-243
localization, 241-243
logic
adding, 181
implementing, 190-196
manifest files, 18, 73-77
naming, 15, 76
networks
design, 199-201
developing, 201-202
packages, 12, 267-270
permissions, managing, 79-81
piracy, 264-265
preferences
accessing, 42-43
configuring, 93-95

Android
Developer websites, 300, 304
Development Tools. See ADTs
SDK
downloading, 282-284
installing, 281-282
Manager, 283, 284
troubleshooting, 287
upgrading, 284
Virtual Device. See AVD
AndroidManifest.xml tab, 18
android.permissionINTERNET permission, 190
android.permissionWAKE_LOCK permission, 203
android.util.log class, 50
animation
applying, 110-115
image settings, 113-114
sequences, 66
splash screens, 101
design, 101
implementing, 102-109
views, 112
APIs (application programming interfaces), 75-76, 228
App Engine (Google), 200
Apple iPhone, 9
Application Manager, 96
Application Nodes section, 78
application programming interfaces. See APIs
<application> tag, 76
Application tab, 71
Application Nodes section, 78

anddev.org, 304
AndroidManifest.xml tab, 73
Android
Developer websites, 300, 304
Development Tools. See ADTs
SDK
downloading, 282-284
installing, 281-282
Manager, 283, 284
troubleshooting, 287
upgrading, 284
Virtual Device. See AVD
AndroidManifest.xml tab, 18
android.permissionINTERNET permission, 190
android.permissionWAKE_LOCK permission, 203
android.util.log class, 50
animation
applying, 110-115
image settings, 113-114
sequences, 66
splash screens, 101
design, 101
implementing, 102-109
views, 112
APIs (application programming interfaces), 75-76, 228
App Engine (Google), 200
Apple iPhone, 9
Application Manager, 96
Application Nodes section, 78
application programming interfaces. See APIs
<application> tag, 76
Application tab, 71
Application Nodes section, 78

applications
activities, implementing, 92-93
attributes, 77
builds, 20-28, 39
activities, 43-45
context, 42-43
dialog boxes, 48-49
fragments, 49-50
intents, 46-48
logging, 50-51
tracking, 74
customizing, 237-240
DDMS, debugging, 31-35
deployment, 12
descriptions, 76
design, 39-41
development, 11
devices, launching, 25-28
DRM, 265
environments, configuring, 279-280
frameworks
design, 83
trivia games, 83-89
functionality
accessing, 44
implementing, 41
Have You Read That?, 84. See also Have You Read That? application
icons, 76
internationalizing, 240-241, 245-246
JDK, installing, 281
launching, 22-25, 47
locales, 241-243
localization, 241-243
logic
adding, 181
implementing, 190-196
manifest files, 18, 73-77
naming, 15, 76
networks
design, 199-201
developing, 201-202
packages, 12, 267-270
permissions, managing, 79-81
piracy, 264-265
preferences
accessing, 42-43
configuring, 93-95
prerelease checklists, 263
projects
  adding resources, 91-92
  creating new, 90-91
prototypes
  implementing, 90-95
  running, 95-97
publishing, 271
registering, 261
release processes, 261-262
resources
  applying, 53-56
  managing, 53, 236
  references, 55-56
  retrieving, 42
  storing, 54
  values, 57-59
servers, applying, 199-200
settings, 81
signing, 267-268
targets, deploying, 27
testing, 249
  best practices, 249-252
  maximizing coverage, 252-260
uploading, 272-273

applying
activities, 43-45
animation, 110-115
applications, servers, 199-200
AsyncTask class, 206
Autocomplete, 293
Button control, 156
colors, 57-58
dialog boxes, 48-49
dimensions, 58-59
drawable resources, 59-60
EditText control, 156
emulators, 36
files, 64-66, 136-137
  raw, 64-65
  XML, 64
fragments, 49-50
GridView control, 124-128
handlers, 207
HTTP, 203-204
images, 59-60
intents, 46-48
layouts, 60-64
logging, 50-51

resources, applications, 53-56
Service objects, 218-219
Spinner control, 159
strings, 57
threads, 207
ViewSwitcher controls, 186-190
XML, 146-147

Appstore (Amazon), 244
arrays, 66
  strings, adding resources, 153
assertTrue() method, 257
AsyncTask class, 206
  applying, 206
  book downloads, extending for, 213
  scores, downloading, 207-208
attributes
  applications, 77, 200
  backgrounds, 122
  inheriting, 136
  inputType, 156
  maxLength, 156
  maxLines, 156
  minLines, 156
  rank, 146
  score, 146
  titles, 129
  username, 146
  XML, TextView control, 136
audio, 64
author’s website, accessing, 301-303
Autocomplete, applying, 293
automation
  builds, 12
  imports, 294. See also imports
  testing, 253-260
available space, 280
AVD (Android Virtual Device), 20
  managing, 20-21

B

backgrounds
  attributes, 122
  managing, 206
  processing, 215
batches of books, parsing, 213-216
best practices, testing applications, 249-252
Bitmap objects, 190
blocks
- code, formatting, 294
try/catch, 137
blogs, 31
BookListDownloaderTask class, starting, 215
books, parsing, 213-216
boolean values, 66
bugs, reporting, 287. See also debugging
builds
- applications, 20-28, 39
  activities, 43-45
  context, 42-43
design, 39-41
dialog boxes, 48-49
  fragments, 49-50
  intents, 46-48
logging, 50-51
testing, 250
automation, 12
errors, troubleshooting, 297
files, 15
processes, 250
release candidates, 263-265
screens, tabs, 144-145
targets, selecting, 13
tracking, 74
Button control, 149
- applying, 156
configuring, 157
items, viewing, 182
buttons
- clicks, 158-159, 193
Copy, 35
Create AVD, 21
Debug, 23
Delete, 33
Menu, 130
Pull, 33
Push, 33
Refresh, 35
Rotate, 34
Screen Capture, 34
Stop Process, 33
Update Heap, 33
Update Threads, 33

callbacks, activities, 44-46
calls, deprecated dialog, 165
cancel() method, 214
candidate builds, releases, 263-265
Canvas, 110
cases
  edge, 195
testing, 255
case statements, 169
CharacterPickerDialog, 166
checking
  network status, 203
for updates, 284
checklists, prerelease, 263
classes
  Activity, 15, 40-41. See also Activity class
customizing dialogs, 171-177
DatePickerDialog class, 168-171
layout files, 40
lifecycles of dialogs, 167-168
methods, 48
android.util.log, 50
AsyncTask, 206
  applying, 206
  downloading scores, 207-208
  extending book downloads, 213
BookListDownloaderTask, starting, 215
Dialog, 165-166
factory, implementing, 187-189
formatting, 293
Fragment, 49
getIntent(), 47
Handler, 206
HashTable, 181, 192
Log, 50
PreferenceFragment, 151
ProgressDialog, 166, 205
ScoreDownloaderTask, 212
Thread, 201, 206
UploaderService, 219
UploadTask, implementing, 220
URL, 203
UUID, 216
controls, buttons, 158-159
driving games forward, 193
clients, 220. See also networks
testing, 251
closing
  activities, 45
tabs, 291
cloud-monitoring solutions, 228
code
  Amazon Web Services SDK, 226
code signing tools, 12
  formatting, 294
  Java, writing, 293-297
  refactoring, 294-296
  reorganizing, 296
  resources, storing, 54
  standards, developing, 250
  versions, settings, 74
collecting user input, 149, 165. See also user input
colors, 54
  applying, 57-58
  resources, adding, 153
commands
  emulators, 36
  Organize Imports, 293
communication protocols, accessing, 199
compatibility, development
  environments, 280
components
  Android SDKs, downloading, 282-284
  games, 181. See also games
configuring
  applications, 81
    manifest files, 73-77
    preferences, 93-95
  AVDs, 20-21
  Button control, 157
  debugging, 22, 286
  development
    devices, 286-287
    environments, 279-280
    machine prerequisites, 280
  EditText control, 156
  manifest files, 69
  network permissions, 203
  resolution, 21
  Spinner control, 160-161
tab defaults, 145
TabHost control, 144
targets, 27
XML, 146-147
conformance, testing, 252
connecting
  devices, 199
  Mac operating systems, 285
  networks, 201. See also networks
  slow connections, 206
  Windows operating systems, 285
consistency of screen design, 133
contacting authors, 303
contains() method, 163
containsKey() method, 193
context
  activities, launching, 43
  applications, 42-43
  menus, 128
controls
  Button, 149
    applying, 156
    configuring, 157
  DatePicker, 166
  EditText, 149, 256
    applying, 156
    configuring, 156
    listening for keystrokes, 175-177
    forms, 155-161
  FrameLayout, 102, 139
  GridView, 117, 237
    adding, 122-123
    applying, 124-128
  ImageButton, 157
  ImageSwitcher, 181-182, 189-190
  ImageView, 60, 102, 118
  interfaces, formatting, 58
  layouts, 66, 102-104
  LinearLayout, 61, 102, 135, 182
  PleaseWaitDialog, 205
  Progressbar, 166, 201, 204-205
  RelativeLayout, 103, 122, 135, 182
  ScrollView, 133, 150
  Spinner, 149, 153
    applying, 159
    configuring, 160-161
  splash screens, 102
controls

TabHost, 142
  adding, 139
  configuring, 144
tabs, 145
TableLayout, 102, 139
TabSpec, 145
TextSwitcher, 181, 182, 189
TextView, 61, 63, 92, 102, 118, 136
TimePicker, 166
View, 181, 182
ViewSwitcher
  applying, 186-190
  implementing factory classes, 187-189
Copy button, 35
copyrights, DRM, 265
costs, 9-12
coverage, maximizing testing, 252-260
Create AVD button, 21
createFromResource() method, 154
creating. See configuring; formatting
currencies, 247
customizing
  applications, 237-240
  layout controls, 66
  log filters, 292

defaults
  layouts, 61
  locales, 242
  resources, 236
  tab settings, 145
defect tracking systems, 251
defining
  activities, 77-79
  color resources, 58
dialogs, 167
help screens, 87
identifier, 168
main menu screen features, 85-86
screens
  game, 88
  scores, 87
  settings, 88
services, 218
SharedPreferences entries, 161
splash screen features, 85
deleteFile() method, 136
deleting dialogs, 168
deployment
  applications, 12, 27
  manual, 25
  releases, 12
deprecated dialog calls, 165
descriptions, applications, 76
design
  applications, 39-41
    frameworks, 83
    networks, 199-201
    trivia games, 83-89
  layouts
    Layout Resource Editor, 61-62
    XML, 62-64
  main menu screens, 117-119
  piracy, 264-265
  prerelease checklists, 263
  screens
    games, 181-183
    help, 133
    scores, 138-139, 142-144
    settings, 149-151
    splash, 101
determinate progress, 204
Developer.com, 304
Developer License Agreement, 272

Dealing with Databases. See DDMS
databases, 199. See also storing
  Amazon Web Services SDK, 226, 227
Date of Birth setting, 149
DatePickerDialog, 166
dates, formatting, 246
DDMS (Dalvik Debug Monitor Service), 97
  applications, debugging, 31-35
  perspective, 25
Debug button, 23
Debug Configuration Manager, 23
debugging, 12, 252, 284-285
  applications, 20-28, 31-35, 95
  configuring, 22, 286
diagnostics, 263
  layouts with XML, 62-64
logging, 50
USB, 280
declaring strings, literals, 192
doInBackground() method
downloaded

Android SDK, 10
batches of books, 213-216
Eclipse, 281
scores, 207-208, 224
SDK, 282-284
tools, 36
drawable resources, 184
adding, 92
applying, 59-60
directories, 61
DDMS, 31 - 35
interaction, 31
launching, 20
managing, 25
networks, testing, 202
prototypes, launching, 95
screenshots, 34-35
SD cards, 20
tabs, viewing, 142
enabling
development settings, 286-287
network support, 201
Snapshot, 21
Ebook readers, 10
Eclipse, 12, 279
ADTs, installing, 282
applications, testing, 253-260
build files, 15
DDMS, 31. See also DDMS
debugging, 20
installing, 281
Layout Resource Editor, 61-62
LogCat, 50
managing, 289-293
Manifest File Resource Editor, 69
manifest files, editing, 17-18
navigating, 12-19
optimizing, 289
preferences, 282
tabs/windows outside of, 291
updating, 284
edge cases, 195
testing, 252
editing
manifest files, 69
resources
files, 18-19
projects, 17-19
SharedPreferences, 162
editors
Layout Resource Editor, 61-62
Manifest File Resource Editor, 73
resources, 12
XML, 18
EditText control, 149, 256
applying, 156
configuring, 156
keystrokes, listening for, 175-177
elements, <score>, 142
Email setting, 149
emulators, 12
applications
launching, 22-25
testing on, 252
applying, 36
DDMS, 31-35
interaction, 31
launching, 20
managing, 25
networks, testing, 202
prototypes, launching, 95
screenshots, 34-35
SD cards, 20
tabs, viewing, 142
enabling
development settings, 286-287
network support, 201
Snapshot, 21
English, 242
entries, defining SharedPreferences, 161
environments
development
configuring, 279-280
IDEs, 280
Eclipse. See Eclipse
JRE, 281
errors
build, troubleshooting, 297
logging, 40
text, viewing, 153
events, 44
GridView control, 127-128
lifecycles, animation, 114
logging, 50
exceptions, NullPointerException, 142
execute() method, 212, 215
existing projects, adding, 15
exporting package files, 267-268
Extensible Markup Language. See XML
Extract Local Variable tool, 295
Extract Method tool, 295
factory classes, implementing, 187-189
features
  applications, networks, 201
  DDMS, 31
design, 39-41
game screens, 182. See also games
  high-level game, 83-84
  main menu screens, defining, 85-86
  tracking, 74
feedback, sending, 303
fields, Minimum SDK, 15
fileList() method, 136
files
  applying, 64-66, 136-137
  file systems, navigating, 33
graphics, 54
  JAR, 281
  jar, 15
  layouts, 54, 60-64
    activity classes, 40
game.xml, 185
settings.xml, 154
XML, 49
main menu screens, updating, 121-123
managing, 31
manifest, 17-18. See also manifest files
  configuring, 69
  launching activities, 43
  navigating, 69-73
  releases, 263-264
packages, exporting, 267-268
projects
  creating, 13-15
  navigating, 15-17
raw, 54, 64-65
raw resource
  accessing, 137
  adding, 136
resources
  editing, 18-19
  XML, 18
sections, viewing, 291
source control, 290
subdirectories, navigating, 18-19

XML
  adding, 58
  applying, 64
  parsing, 146-147
filling GridView controls, 124-127
filters, customizing logs, 292
findViewByld() method, 64, 137, 145
finish() method, 45, 115
FierceDeveloper, 304
folders, source code, 15
formatting
  applications, creating new projects, 90-91
  classes, 293
code, 294
  colors, 57-58
  files, 64-66
graphics, 60
  interface controls, 58
  layouts, 60-64
  methods, 293
  projects, 13-15
settings screens, 149-151
strings, 57, 246
tab defaults, 145
XML, 146-147
forms, controls, 155-161
forums, 10
Fragment class, 49
fragments
  applying, 49-50
  dialogs, 49
frame-by-frame animation, 110
FrameLayout control, 102, 139
frameworks, applications
  design, 83
  trivia games, 83-89
French, 242
frequency of testing, 253
full applications, implementing
  internationalization, 245-246
functionality
  applications
    accessing, 44
    implementing, 41
  networks, 201
testing, 251
top-level application, 42
G

game.xml layout file, 185
games. See also applications
  logic, implementing, 190-196
  options menus, adding, 129-130
  running, 95-97
screens
  defining, 88
  design, 181-183
  implementing, 183-186
  updating, 239
settings, 149
trivia, 83-89
garbage collection, 33
Gender setting, 149
generating reports, Amazon Appstore, 274
German, 242
GET method, 220-223
get() method, 193
getApplicationContext() method, 42
getAttributeValue() method, 147
getBookCoverImageDrawable() method, 190
getColor() method, 58
getDimension() method, 59
getDir() method, 136
getDrawable() method, 60
getFilesDir() method, 136
getFileStreamPath() method, 136
getFormat() method, 163
getIntent() method, 47, 128
getLong() method, 163
getResources() method, 42, 56
getString() method, 57, 163
getText() method, 156
getView() method, 126
GIF (Graphics Interchange Format), 60, 110
Google
  App Engine, 200
  Mobile Services, 253
  Open Handset Alliance, 9-10
  Plus Android Developers, 304
  TV, 10
graphical layout editors, 12
Graphical Layout tab, 123
graphics, 53
  adding, 120
  animation, 110
files, 54
formatting, 60
subdirectories, 19
Graphics Interchange Format. See GIF
GridView control, 117, 237
  adding, 122-123
  applying, 124-128
  guidelines, testing applications, 250

H
handleNoBooks() method, 196
Handler class, 206
handlers, applying, 207
Hashtable class, 181
  books, storing, 192
Have You Read That? application, 84
  game screens, 181
  help screens, 133-137
  main menu screens, 118
  menus, 128
  network support, 202
  options menus, 130
  scores screens, 133, 138-144
  servers, 200
  settings screen passwords, 172
  splash screens, 101, 126
  user input, collecting, 149
headers
  main menu screens, 118
  scores screens, adding, 147
HelloKindleDebug configuration, 23, 25
help screens, 40, 133
  defining, 87
  design, 133
  layouts
    implementing, 135-136
    updating, 135-136
HelloActivity class, 40, 47
hierarchies, alternative, 233-236
Hierarchy Viewer, 12
high-level game features, 83-84
HTML (Hypertext Markup Language), 29
HTTP (Hypertext Transfer Protocol), 199-200, 203-204
HttpClient package, 220
HTTPS (Hypertext Transfer Protocol Secure), 199
Hypertext Markup Language. See HTML
input
Activity class dialogs, 165-168
   customizing, 171-177
   DatePickerDialog class, 168-171
   lifecycles, 167-168
emulators, 36
   form controls, 155-161
   passwords, 172
settings screens, implementing, 151-155
SharedPreferences, 161-163
text, 156
users, collecting, 149
inputStreamToString() method, 137
inputType attribute, 156
insertScoreRow() method, 147
installing
   ADTs, 282
   Android SDK, 281-282
   Eclipse, 281
   JDK, 281
   signed application packages, 269
instrumentation, manifest files, 18
Instrumentation tab, 72
Int object, 44
integers
   arrays, 66
   values, 66
integrated development environments. 
   See IDEs
integrating
   source-control services, 289
   testing, 251
<intent-filter> tag, 79
Intent information, 130
intents, applying, 46-48
interfaces, 12, 53
   controls, formatting, 58
   Eclipse, managing, 289-293
   layout files, 19
   Layout Resource Editor, 61-62
   orienting, 233
   settings.xml layout file, 154
tivia games, 83-84
internationalization, 54
   applications, 240-241
   full applications, implementing, 245-246
   strategies, 243-246
Internet, 199. See also networks
resources, 304
service providers. See ISPs
slow connections, 206
ISO 639-1, 242
ISPs (Internet service providers), 227
<item> tag, 129
items, viewing, 182
iteration, loops, 146

J

JAR files, 15, 281
Java, 11
code, writing, 293-297
Development Kit. See JDK
documentation, Amazon Web Services
SDK, 226
resource layouts, 241
Runtime Environment. See JRE
Server Pages. See JSP
Java documentation, 297
JDK (Java Development Kit), 279, 281
Joint Photographic Experts Group. See JPEG
JPEG (Joint Photographic Experts Group), 60
JRE (Java Runtime Environment), 281
JSP (JavaServer Pages), 200, 207
batch requests, 213-216
JUnit, 253-260

K

keyboard input, emulators, 36
keystrokes, listening for, 175-177
Kindle Fire. See devices

L

landscape mode, 237
languages
HTML, 29
Java, 11
localization, 240-243
specific resources, 242
support, 233
XML, 18
launching
activities, 43-44, 78-79
applications, 22-25
applying intents, 47
devices, 25-28
custom password dialogs, 177
DatePickerDialogs, 170
dialogs, 167
emulators, 20
icons, 92
prototypes, 95
Layout Resource Editor, 61-62
layouts
applying, 60-64
controls, 66, 102-104
design
help screens, 133
Layout Resource Editor, 61-62
XML, 62-64
files, 54
activity classes, 40
game.xml, 185
settings.xml, 154
XML, 49
game screens, 183. See also games
help screens
implementing, 135-136
updating, 135-136
main menu screens, 117-119, 121-123
password dialogs, implementing, 174
resources
adding, 92
Java, 241
programming, 63
scores screens, 138
implementing, 141-144
updating, 142-144
settings screens, 151-155
splash screens, updating, 106-109
LBS (Location-Based Services), 201
leaks, managing, 250
libraries
Amazon Web Services SDK, 226
LVL, 253
License Verification Library. See LVL
licenses, Developer License Agreement, 272
lifecycles
animation, 114
callback activities, 46
Fragment class, 49
managing
activities
dialogs, 48
state, 44-45
Amazon Web Services SDK, 226, 228
Application Manager, 96
application resources, 53, 236
AVDs, 20-21
backgrounds, 206
Debug Configuration Manager, 23
DRM, 265
Eclipse, 289-293
emulators, 25
files, 31
imports, 293-294
memory, 31, 250
networks, 200
qualifiers, 234-236
resources, 19, 104, 233-236
tasks, 31-32
upgrading, 74
windows, 291
Manifest tab, 70
<manifest> tag, 74
manual deployment, 25
master layouts, 122. See also layouts
maximizing
testing coverage, 252-256
windows, 290
maxLength attribute, 156
maxLines attribute, 156
measurements, dimensions, 59
media files, resources, 65

limitations
of emulators, 36
of internationalization, 244, 245
LinearLayout control, 61, 102, 139, 150
Linux operating systems, 280
listening for events, 127
literals, declaring strings, 192
locales
adding, 241
applications, 241-243
devices, 243
settings, 241
systems, 246
localization, 54, 240-241
applications, 241-243
tools, 246-247
Location-Based Services. See LBS
locations, source code, 13
Log class, 50
LogCat, 12, 35, 50
navigating, 292
logging, 31
applying, 50-51
viewing, 35
logic, applications
adding, 181
implementing, 190-196
logs, customizing filters, 292
loops, while(), 146
LVL (License Verification Library), 253, 264

Mac operating systems, 280
connecting, 285
debugging, 286
machine prerequisites, 280
main menu screens
defining, 85-86
design, 117-119
GridView control, applying, 124-128
implementing, 117
resources, adding, 120-121
types, 128-130
updating, 121-123, 237-238
main.xml tab, 61
maintenance, 35
makeView() method, 187

manifest files, 17-18
activities
defining, 77-79
launching, 43
applications
attributes, 77
configuring, 73-77
descriptions, 76
icons, 76
managing permissions, 79-81
naming, 76
settings, 81
configuring, 69
navigating, 69-73
releases, 263-264
updating, 93
Manifest tab, 70
<manifest> tag, 74
managing
dialogs, 48
state, 44-45
Amazon Web Services SDK, 226, 228
Application Manager, 96
application resources, 53, 236
AVDs, 20-21
backgrounds, 206
Debug Configuration Manager, 23
DRM, 265
Eclipse, 289-293
emulators, 25
files, 31
imports, 293-294
memory, 31, 250
networks, 200
qualifiers, 234-236
resources, 19, 104, 233-236
tasks, 31-32
upgrading, 74
windows, 291
Manifest tab, 70
<manifest> tag, 74
manual deployment, 25
master layouts, 122. See also layouts
maximizing
testing coverage, 252-256
windows, 290
maxLength attribute, 156
maxLines attribute, 156
measurements, dimensions, 59
media files, resources, 65
memory, managing, 31, 250
Menu button, 130
<menu> tag, 129
MenuActivity class, 40, 44
menus, 40, 66
  context, 128
  options, 128-130
  types of, 128-130
messages
  Amazon Web Services SDK, 226-228
  logging, 50
  privacy, 202
methods
  addTab(), 145
  assertTrue(), 257
  callbacks, 44-46
  cancel(), 214
  contains(), 163
  containsKey(), 193
  createFromResource(), 154
  deleteFile(), 136
  dismiss(), 205
  dismissDialog(), 48, 167
  doIn Background(), 206, 210-211, 220
  execute(), 212, 215
  fileList(), 136
  findViewByld(), 64, 137, 145
  finish(), 45, 115
  formatting, 293
  GET, 220-223
  get(), 193
  getApplicationContext(), 42
  getAttributeValue(), 147
  getBookCoverImageDrawable(), 190
  getColor(), 58
  getDimension(), 59
  getDir(), 136
  getDrawable(), 60
  getFilesDir(), 136
  getFileStreamPath(), 136
  getIntent(), 163
  handleNoBooks(), 196
  handleNoBooks(), 196
  inputStreamToString(), 137
  insertScoreRow(), 147
  Log class, 50
  makeView(), 187
  next(), 146
  onActivityResult(), 44
  onBind(), 219
  onCancelled(), 209
  onCreate(), 44, 78, 125, 161
  Hashtable class, 193
  initializing switchers, 187
  onCreateDialog(), 48, 167, 168
  onCreateOptionsMenu(), 129
  onCreateView(), 49
  onDestroy(), 44, 49
  onDestroyed(), 163
  onFinished(), 259
  onClick(), 128
  onCreateOptionsMenu(), 130
  onOptionsItemSelected(), 130
  onPause(), 44, 49, 163, 219
  onPickDateButtonClick(), 158
  onPreExecute(), 206, 208, 214
  onPrepareDialog(), 48, 167, 168
  onProgressUpdate(), 206, 211
  onResume(), 44
  onSetPasswordButtonClick(), 158
  onStart(), 49
  onStartCommand(), 219
  onStop(), 49
  openFileInput(), 136
  openRawResource(), 64
  POST, 220
  put(), 193
  putExtra(), 47
  randomUUID(), 216
  removeDialog(), 48, 167, 168
  setContent(), 145
  setContentView(), 49, 125
  setCurrentTabByTag(), 145
  setCurrentText(), 189
  setFactory(), 187
  setImageDrawable(), 189
  setInput(), 203
  setProgress(), 189
  setSelection(), 160
  setText(), 156
  setUp(), 256
remote servers, uploading settings data to, 217-223
resources, accessing, 80
scores
downloading, 208-213, 224
uploading, 223
servers, 141
services, accessing, 202-204
status, checking, 203
support, 199, 202
tasks, running asynchronously, 206-207
New Android Project dialog box, 15
new projects, creating, 13-15
next() method, 146
Nickname setting, 149
nine-patch stretchable images, 60
notifications, 44
Amazon Web Services SDK, 227-228
NullPointerException, 142
obfuscation, 12
objects
Bitmap, 190
Int, 44
releases, 250
Service, applying, 218-219
onActivityResult() method, 44
onBind() method, 219
onCancelled() method, 209
onCreate() method, 44, 78, 125, 161
Drawable class, 193
switchers, initializing, 187
onCreateDialog() method, 48, 167, 168
onCreateOptionsMenu() method, 130
onCreateView() method, 49
onDestroy() method, 44, 49
onDestroyed() method, 163
onFinish() method, 259
onItemClick() method, 128
onPostExecute() method, 206, 208
progress dialogs, dismissing, 214
progress dialogs, starting, 214

setup(), 145
showDialog(), 48, 167
startActivity(), 44, 46
toString(), 156
updateDate(), 170
minimizing windows, 290
Minimum SDK field, 15
minimum SDK versions, 75-76
minLines attribute, 156
mobile devices, 199. See also devicesmock
score data, 141. See also scores
modes
docking, 233
landscape, 237
money, 247
monitoring, 12, 228
moving tabs, 290
MySQL, 200

N
naming
applications, 76
packages, 74
projects, 13
versions, 74
navigating
Amazon Web Services SDK, 226-228
Eclipse, 12-19
emulators, 36
file systems, 33
manifest files, 69-73
project files, 15-17
prototypes, 96-97
subdirectories, 18-19
networks
activities, 201
applications
design, 199-201
developing, 201-202
batches of books, downloading, 213-216
emulators, testing, 202
Kindle Fire, testing, 202
managing, 200
permissions, configuring, 203
player data
synchronizing, 216-217
uploading, 216
ProgressBar control, 204-205

O
obfuscation, 12
objects
Bitmap, 190
Int, 44
releases, 250
Service, applying, 218-219
onActivityResult() method, 44
onBind() method, 219
onCancelled() method, 209
onCreate() method, 44, 78, 125, 161
Drawable class, 193
switchers, initializing, 187
onCreateDialog() method, 48, 167, 168
onCreateOptionsMenu() method, 130
onCreateView() method, 49
onDestroy() method, 44, 49
onDestroyed() method, 163
onFinish() method, 259
onItemClick() method, 128
onPostExecute() method, 206, 208
progress dialogs, dismissing, 214
progress dialogs, starting, 214
onPrepareDialog() method, 48, 167, 168
onProgressUpdate() method, 206, 211
onResume() method, 44
onSetPasswordButtonClick() method, 158
onStart() method, 49
onStartCommand() method, 219
onStop() method, 49
Open Handset Alliance, 9-10
open platforms, 9
openFileDialog() method, 136
OpenGL, 110
OpenIntents, 304
openRawResource() method, 64
operating systems
  debugging, configuring, 286
  support, 280
optimizing
code, 12
Eclipse, 289
scores screens, 147
options
  adding, 129-130
  menus, 128
org.apache.http.client package, 220
Organize Imports command, 293
orienting interfaces, 233

P

packages
  applications, 12, 267-270
  files, exporting, 267-268
  HttpClient, 220
  naming, 15, 74
  org.apache.http.client, 220
panes, Logcat, 292
parameters
  JSP pages, 213
  View, 158
parsing
  book data, 192-196
  books, 213-216
  XML
    declaring string literals, 192
    files, 146-147
passing information with intents, 47
Password setting, 149

passwords
  colors, 153
  dialogs
    launching, 177
    layouts, implementing, 174
    settings screens, 172
performance, testing, 252
permissions
  android.permission.INTERNET, 190
  android.permission.WAKE_LOCK, 203
  applications, managing, 79-81
  manifest files, 18
  networks, configuring, 203
Permissions tab, 71
perspectives, repositioning tabs, 290
PHP, 200
Pick Date controls, 158
piracy, protecting applications from, 264-265
pixels, configuring, 21
planning
  applications, testing, 251-252
  network support, 202
  publishing, 261
platforms
  Open Handset Alliance, 9
  overview of, 10-11
  resources, 54
play screens, 40
PlayActivity class, 40
player data
  synchronizing, 216-217
  uploading, 216, 220-223
pleaseWaitDialog control, 205
plug-ins, ADTs, 280. See also ADTs
  installing, 282
PNG (Portable Network Graphics), 60
Portable Network Graphics. See PNG
  ports, HTTP, 203. See also connecting
  positioning tabs, 290
POST method, 220
precedence, events, 44
PreferenceFragment class, 151
preferences
  applications
    accessing, 42-43
    configuring, 93-95
  Eclipse, 282
  retrieving, 220
  SharedPreferences, 161-163
prerelease checklists, 263
primitives, 66
privacy messages, 202
processes
  backgrounds, 215
  builds, 250
  releases, 261-262
programming
  alternative resources, 236
  development machine prerequisites, 280
  image resources, 60
  resources
    layouts, 63
    storing, 54
ProgressBar control, 166, 201
  network activities, 204-205
ProgressDialog class, 166, 205
ProGuard, 12, 15, 264-265
projects
  applications
    adding resources, 91-92
    creating new, 90-91
  creating, 13-15
  existing, adding, 15
  files, navigating, 15-17
  resources
    adding, 135
    editing, 17-19
    searching, 292
    testing, 254
property animation, 110
protocols, communication, 199
prototypes
  applications
    implementing, 90-95
    running, 95-97
    launching, 95
publisher's websites, accessing, 300
publishing
  Amazon AppStore, selling on, 271-274
  applications, 271
  planning, 261
  royalties, 273
publishProgress() method, 206
Pull button, 33
Pull Parser (XML), 146
Push button, 33
put() method, 193
putExtra() method, 47
Q
qualifiers, managing, 234-236
queries, application servers, 200
QuickFix, 296-297
R
randomUUID() method, 216
rank attribute, 146
rankings, 138. See also scores
raw files, 54
  applying, 64-65
  XML, editing out, 63
raw resource files
  accessing, 137
  adding, 136
raw sockets, 199
refactoring code, 294-296
references, 31
  application resources, 55-56
Refresh button, 35
regions, resources, 242-243
registering
  accounts, 266
  activities, 77-78
  applications, 261
  developers
    accounts, 271
    Amazon Appstore, 266
  release processes, 261-262
RelativeLayout control, 103, 122, 135, 182
releases
  candidate builds, 263-265
  deployment, 12
  manifest files, 263-264
  objects, 250
  prerelease checklists, 263
  processes, 261-262
  services, 265
remote access. See also accessing
remote servers, 200
  scores, uploading, 223
  settings, uploading, 217-223
removeDialog() method, 48, 167, 168
removing dialogs, 168
Rename tool, 294
RenderScript, 110
reorganizing code, 296. See also managing
reporting bugs, 287
reports, Amazon Appstore, 274
repositioning tabs, 290
requests
   batches, JSP, 213-216
   HTTP, 200
   resources, 234
requirements
   help screens, 133
   main menu screens, 118
   scores screens layouts, 138
resetting, 290. See also settings
resolution, configuring, 21
Resource Chooser, 123
resources, 31
   adding, 91-92
   applications
      applying, 53-56
      managing, 53, 236
      references, 55-56
   retrieving, 42
   storing, 54
   values, 57-59
colors, adding, 153
dimensions, 184
drawable
   adding, 92
   applying, 59-60
directories, 61
editors, 12
files
   editing, 18-19
   XML, 18
images, programming, 60
layouts, 60-64
   Java, 241
   programming, 63
main menu screens, adding, 120-121
managing, 19, 104, 233-236
networks, accessing, 80
online, 304
projects
   adding, 135
   editing, 17-19
raw files, adding, 136-137
regions, 242-243
releases, 250
requests, 234
scores screens, 141-142
settings screens, 151
splash screens, adding, 103-106
strings, adding, 153
types of, 66
URIs, 48
XML, retrieving, 146
responding to events, 127
results, launching activities, 44
retrieving
   application resources, 42
   book data, 192-196
   preferences, 42, 220
   shared preferences, 94
   XML resources, 146
reviewing
   manifest files, 263
   source code, 90
RIM Blackberry, 9
Rotate button, 34
routines, parsing, 146
royalties, publishing, 273
rules, alternative resources, 234
running. See also builds
   applications, 20-28
   automated tests, 258
   configuring, 22
   prototypes, 95-97
   tasks asynchronously, 206-207
runtime
   JRE, 281
   resources, accessing, 53
saving
   activity state, 45
   preferences, 42
   settings to SharedPreferences, 161
SAX (Simple API for XML), 146
score attribute, 146
<score> element, 142
score screens, 40
<score> tag, 146
ScoreDownloaderTask class, 212
scores
downloading, 207-208, 224
screens, 133
  defining, 87
  design, 138-139
  implementing, 141-144
  layouts, 138
  optimizing, 147
  updating, 142-144
servers, uploading, 223
trivia games, 83
viewing, 208-213
ScoresActivity class, 40
Screen Capture button, 34
screens. See also interfaces
dialogs, 48-49. See also dialog boxes
games
  defining, 88
  design, 181-183
  implementing, 183-186
  updating, 239
help, 133
  defining, 87
  design, 133
main menu
defining, 85-86
design, 117-119
implementing, 117
updating, 237-238
scores, 133
  defining, 87
  design, 138-139
  implementing, 141-144
  layouts, 138
  optimizing, 147
  updating, 142-144
settings
customizing dialogs, 172-175
defining, 88
design, 149-151
implementing layouts, 151-155
passwords, 172
splash, 40
  trivia games, 85-86
  updating, 238-239
tabs, builds, 144-145
  trivia games, 83-84
screenshots, 34-35
scrolling, 133. See also navigating
ScrollView control, 133, 150
SD cards, emulators, 20
SDK (Software Development Kit), 9, 279
Amazon Web Services SDK, 225
  navigating, 226-228
  overview of, 225-226
documentation, 30
downloading, 282-284
installing, 281-282
Minimum SDK field, 15
preview versions, 10
qualifiers, applying, 236
targets, selecting, 13
troubleshooting, 287
upgrading, 284
versions, 75-76
searching projects, 292
sections, viewing files, 291
security
  permissions, 79-81
  piracy, 264-265
selecting
  builds, targets, 13
  options menus, 130
  skins, 21
selling on Amazon Appstore, 271-274.
  See also publishing
sequences, animation, 66
servers
  applications, applying, 199-200
  networks, 141
remote, uploading settings data to, 217-223
scores, uploading, 223
testing, 251
Service objects, applying, 218-219
services
  Amazon Web Services SDK, 225
    navigating, 226-228
    overview of, 225-226
  DRM, 265
  networks, 201-204
  releases, 265
  source-control, integrating, 289
UploaderService class, 219
SES (Amazon Simple Email Service), 227
set-top boxes, 10
setContent() method, 145
setContentView() method, 49, 125
setCurrentTabByTag() method, 145
setCurrentText() method, 189
setFactory() method, 187
setImageDrawable() method, 189
setInput() method, 203
setProgress() method, 204
setSelection() method, 160
setText() method, 156
settings
  animation, images, 113-114
  applications, 81
  code, formatting, 294
  games, adding state, 191
  locales, 241
  manifest files, 73-77
  options menus, 128
  perspectives, 290
  remote servers, uploading, 217-223
screens
  customizing dialogs, 172-175
  defining, 88
  design, 149-151
  implementing layouts, 151-155
  passwords, 172
  tabs, defaults, 145
  version code, 74
  XML, 146-147
Settings icon, 202
settings.xml layout file, 154
setUp() method, 145, 256
SharedPreferences, 161-163
  game state settings, adding, 191
sharing preferences, retrieving, 94
showDialog() method, 48, 167
shutting down activities, 45
side-by-side, viewing windows, 291
signing applications, 267-268
Simple API for XML. See SAX
simulating devices, 252
sizing controls, 58
skins, selecting, 21
slow connections, Internet, 206
smartphones, 10
Snapshot, enabling, 21
SNS (Amazon Simple Notification Service), 227
software system requirements, 280. See also applications
source code, 299-300. See also code
  folder, 15
  locations, 13
  reviewing, 90
source-control services, integrating, 289
Spinner control, 149, 153
  applying, 159
  configuring, 160-161
splash screens, 40
  animation, 101
  applying, 110-115
  design, 101
  implementing, 102-109
  controls, 102
  Have You Read That? application, 126
  layouts, updating, 106-109
  resources, adding, 103-106
  trivia games, 85-86
  updating, 238-239
SplashActivity class, 40
SQS (Amazon Simple Queue Service), 228
standards, 9
  applications, testing, 250
  code, developing, 250
startActivity() method, 44, 46
starting
  BookListDownloaderTask class, 215
  progress dialogs, 214
state
  activities, managing, 44-45
  games, adding settings, 191
  networks, 203
statements
  case, 169
  switch, 169
status, checking networks, 203
Stop Process button, 33
storing
  Amazon Web Services SDK, 227
  application resources, 54
  book data, 192-196
strategies, internationalization, 243-246
String resources, 184
strings, 53-54
  applying, 57
  arrays, 66
formatting, 246
languages, specifying, 242
literals, declaring, 192
resources, adding, 91, 153
scores screen, 141
styles, 66. See also formatting
Javadoc documentation, 297
subclasses, 60. See also classes
subdirectories
docs, 29-31
navigating, 18-19
support
colors, formatting, 58
devices, 233
images, 59-60
languages, 233
networks, 199, 202
operating systems, 280
ProGuard, 12
tabs, 139
versions, 15
XML, 146-147
switch statements, 169
switchers, initializing, 187
synchronizing player data, 216-217
systems
default tracking, 251
locales, 246
requirements
development machine
prerequisites, 280
environments, 280
resources, applying, 53-56
Graphical Layout, 123
Instrumentation, 72
main.xml, 61
Manifest, 70
manifest editor tabs, 17
Manifest File Resource Editor, 69
outside of Eclipse, 291
Permissions, 71
perspectives, repositioning, 290
scores, 208
screens, builds, 144-145
support, 139
viewing, 142
TabSpec control, 145
tags
<application>, 76
<item>, 129
<manifest>, 74
<intent-filter>, 79
<score>, 146
<scores>, 146
<uses-permission>, 80, 203
targets
applications, deployment, 27
builds, selecting, 13
devices, testing on, 253
tasks
Eclipse, managing, 292-293
managing, 31, 32
networks, running asynchronously, 206-207
TCP/IP (Transmission Control Protocol/Internet Protocol), 199
templates, 61. See also layouts
GridView control, adding, 122-123
main menu screens, 121
testing
applications, 249
best practices, 249-252
maximizing coverage, 252-260
packages, 268-270
eedge-case, 252
networks
emulators, 202
Kindle Fire, 202
prereleases, 263
projects, 15
TabHost control, 142
adding, 139
configuring, 144
tabs, adding, 145
TableLayout control, 102, 139
tables, 10
tabs
adding, 145
AndroidManifest.xml, 73
Application, 71
Application Nodes section, 78
closing, 291
default settings, 145
errors, viewing, 153
input, 156
TextSwitcher control, 181-182
updating, 189
TextView control, 61, 63, 92, 102, 118
XML attributes, 136
themes, 66
Thread class, 201, 206
threads, applying, 207
time, formatting, 246
TimePickerDialog, 166
titles, attributes, 129
T-Mobile G1, 10
tools, 9, 282
ADTs, installing, 282
Android Developer websites, 300
code signing, 12
development, 29
DDMS, 31-35
documentation, 29-31
Device Screen Capture, 34
downloading, 36, 282-284
emulators. See emulators
Extract Local Variable, 295
Extract Method, 295
Hierarchy Viewer, 12
localization, 246-247
LogCat, 12, 50
Rename, 294
upgrading, 284
XML, 64
top-level application functionality, 42
toString() method, 156
tracking
builds, 74
defect tracking systems, 251
features, 74
transitions, activities, 165
Transmission Control Protocol/Internet Protocol. See TCP/IP
trivia games, 83-89
running, 95-97
troubleshooting, 35, 282
Android SDK, 287
build errors, 297
cloud-monitoring solutions, 228
defect tracking systems, 251
main screens, 237
networks, 203
QuickFix, 296-297
slow connections, 206
try/catch blocks, 137
typed arrays, 66
types
of animation, 110
of dialogs, 166-167
of drawable resources, 60
of main menu screens, 128-130
of resources, 53, 66
of scores, 138

uniform resource identifiers. See URIs
Update Heap button, 33
Update Threads button, 33
updateDate() method, 170
upgrading
Android SDK, 284
managing, 74
UploaderService class, 219
uploading
applications, 272-273
player data, 216, 220-223
scores, servers, 223
settings, remote servers, 217-223
UploadTask class, implementing, 220
URIs (uniform resource identifiers), 48
URL class, 203
usability testing, 251
USB (universal serial bus), debugging, 280
user input
Activity class dialogs, 165-168
customizing, 171-177
DatePickerDialog class, 168-171
lifecycles, 167-168
types of, 166-167
collecting, 149
dialog boxes, 165
form controls, 155-161
settings screens
design, 149-151
implementing, 151-155
SharedPreferences, 161-163
username attribute, 146
<users-permission> tag, 80, 203
utilities. See tools
UUID class, 216

V
validating identifiers, 216
values
application resources, 57-59
boolean, 66
integers, 66
preferences, 94
verifying
LVL, 253
manifest file releases, 263
signed applications, 269
versions
applications, testing, 250
builds, 250
Eclipse, 281
manifest files, 74-75
SDK, 75-76
support, 15
videos, 31
View controls, 181, 182
View parameter, 158
viewing
eroerror text, 153
file sections, 291
items, 182
logging, 35
progress, 204-205
scores, 208-213
tabs, 142
windows side-by-side, 291
views, animation, 110, 112
ViewSwitcher controls
applying, 186-190
factory classes, implementing, 187-189

W
warnings, logging, 50
websites
Android Developer, 300, 304
author's, accessing, 301-303
developers, 282
InformIT, 301
publisher's, accessing, 300
while() loops, 146
Wi-Fi, 202. See also connecting
permissions, 203
windows
managing, 291
maximizing, 290
minimizing, 290
outside of Eclipse, 291
side-by-side, viewing, 291
Windows operating systems, 280
connecting, 285
debugging, 286
Wireless Developer Network, 304
wizards, Android Project, 12-13
workspaces, Eclipse, 289-293
writing code, Java, 293-297

X–Z
XML (Extensible Markup Language), 18
animation, 111. See also animation
attributes, TextView control, 136
files
adding, 58
applying, 64
parsing, 146-147
game screens, implementing, 184-185
layouts
design, 62-64
files, 49, 60-64
parsing, declaring string literals, 192
Pull Parser, 146
resources, retrieving, 146
support, 146-147
XmlResourceParser, 146-147