Lauren Darcey
Shane Conder

Third Edition
Features Android 4.0

Android™ Wireless Application Development

Volume I: Android Essentials

Developer's Library

Android™ Wireless Application Development

Volume 1: Android Essentials

Third Edition

This page intentionally left blank

Android™ Wireless Application Development

Volume 1: Android Essentials

Third Edition

Lauren Darcey
Shane Conder

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

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

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
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data:
Darcey, Lauren, 1977-
Android wireless application development / Lauren Darcey, Shane Conder.
v. cm.
Includes bibliographical references and index.
Contents: v. 1. Android essentials
ISBN 978-0-321-81383-1 (pbk. : alk. paper)
1. Application software–Development. 2. Android (Electronic resource) 3. Mobile computing. I. Conder, Shane, 1975- II. Title.
QA76.76.A65D258 2012b
005.1–dc23
2011049390

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.

Android is the trademark of Google, Inc. Pearson Education does not assert any right to the use of the Android trademark and neither Google nor any other third party having any claim in the Android trademark have sponsored or are affiliated with the creation and development of this book.

Some figures that appear in this book have been reproduced from or are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License (http://creativecommons.org/licenses/by/2.5/).

ISBN-13: 978-0-321-81383-1
ISBN-10: 0-321-81383-9

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

First Printing: February 2012

Editor-in-Chief
Mark Taub

Acquisitions Editor
Trina MacDonald

Development Editor
Songlin Qiu

Managing Editor
Kristy Hart

Project Editor
Lori Lyons

Copy Editor
Bart Reed

Indexer
Heather McNeill

Proofreader
Paula Lowell

Technical Reviewers
Douglas Jones
Mike Wallace
Mark Gjoel

Publishing Coordinator
Olivia Basegio

Multimedia Developer
Dan Scherf

Book Designer
Gary Adair

Compositor
Nonie Ratcliff

❖

This book is dedicated to Chickpea.

❖

Contents at a Glance

Introduction 1

I: An Overview of the Android Platform

1 Introducing Android 11

2 Setting Up Your Android Development Environment 37

3 Writing Your First Android Application 53

4 Mastering the Android Development Tools 83

II: Android Application Basics

5 Understanding the Anatomy of an Android Application 103

6 Defining Your Application Using the Android Manifest File 119

7 Managing Application Resources 137

III: Android User Interface Design Essentials

8 Exploring User Interface Screen Elements 171

9 Designing User Interfaces with Layouts 199

10 Working with Fragments 233

11 Working with Dialogs 251

IV: Android Application Design Essentials

12 Using Android Preferences 263

13 Working with Files and Directories 275

14 Using Content Providers 285

15 Designing Compatible Applications 301

V: Publishing and Distributing Android Applications

16 The Android Software Development Process **325**

17 Designing and Developing Bulletproof Android Applications **347**

18 Testing Android Applications **363**

19 Publishing Your Android Application **377**

VI: Appendixes

A The Android Emulator Quick-Start Guide **399**

B The Android DDMS Quick-Start Guide **423**

C Eclipse IDE Tips and Tricks **439**

Index **449**

Table of Contents

Introduction 1

- Who Should Read This Book 1
- Key Questions Answered in This Volume 2
- How These Books Are Structured 2
- An Overview of Changes in This Edition 4
- Development Environment Used in This Book 5
- Supplementary Materials Available 6
- Where to Find More Information 6
- Conventions Used in This Book 7
- Contacting the Authors 8

I: An Overview of the Android Platform

1 Introducing Android 11

- A Brief History of Mobile Software Development 11
  - Way Back When 11
  - “The Brick” 13
  - Wireless Application Protocol (WAP) 15
  - Proprietary Mobile Platforms 17
- The Open Handset Alliance 19
  - Google Goes Wireless 19
  - Forming the Open Handset Alliance 19
  - Manufacturers: Designing Android Devices 20
  - Mobile Operators: Delivering the Android Experience 21
  - Apps Drive Device Sales: Developing Android Applications 22
  - Taking Advantage of All Android Has to Offer 22
  - The Android Marketplace: Where We’re at Now 22
- Android Platform Differences 23
  - Android: A Next-Generation Platform 24
  - Free and Open Source 25
  - Familiar and Inexpensive Development Tools 25
  - Reasonable Learning Curve for Developers 26

Enabling Development of Powerful Applications 26
Rich, Secure Application Integration 26
No Costly Obstacles to Publication 27
A “Free Market” for Applications 27
A Growing Platform 28
The Android Platform 29
Android’s Underlying Architecture 29
Security and Permissions 31
Developing Android Applications 32
Summary 35
References and More Information 35

2 Setting Up Your Android Development Environment 37
Configuring Your Development Environment 37
Configuring Your Operating System for Device Debugging 39
Configuring Your Android Hardware for Debugging 39
Upgrading the Android SDK 41
Problems with the Android Software Development Kit 41
Exploring the Android SDK 42
Understanding the Android SDK License Agreement 42
Reading the Android SDK Documentation 43
Exploring the Core Android Application Framework 43
Exploring the Core Android Tools 46
Exploring the Android Sample Applications 50
Summary 52
References and More Information 52

3 Writing Your First Android Application 53
Testing Your Development Environment 53
Adding the Snake Project to Your Eclipse Workspace 54
Creating an Android Virtual Device (AVD) for Your Snake Project 56

Creating a Launch Configuration for Your Snake Project 58

Running the Snake Application in the Android Emulator 59

Building Your First Android Application 62

Creating and Configuring a New Android Project 62

Core Files and Directories of the Android Application 65

Creating an AVD for Your Project 65

Creating a Launch Configuration for Your Project 66

Running Your Android Application in the Emulator 67

Debugging Your Android Application in the Emulator 69

Adding Logging Support to Your Android Application 73

Adding Some Media Support to Your Application 74

Debugging Your Application on the Hardware 78

Summary 80

References and More Information 81

4 Mastering the Android Development Tools 83

Using the Android Documentation 83

Leveraging the Android Emulator 85

Viewing Application Log Data with LogCat 86

Debugging Applications with DDMS 87

Using Android Debug Bridge (ADB) 87

Using the Resource Editors and UI Designer 88

Using the Android Hierarchy Viewer 91

Launching the Hierarchy Viewer 92

Working in Layout View Mode 92

Optimizing Your User Interface 94

Working in Pixel Perfect Mode 94

Working with Nine-Patch Stretchable Graphics 95

Working with Other Android Tools 98

Summary 99

References and More Information 100

II: Android Application Basics

5 Understanding the Anatomy of an Android Application 103
Mastering Important Android Terminology 103
Using the Application Context 104
Retrieving the Application Context 104
Using the Application Context 104
Performing Application Tasks with Activities 106
The Lifecycle of an Android Activity 106
Organizing Activity Components with Fragments 111
Managing Activity Transitions with Intents 113
Transitioning Between Activities with Intents 113
Organizing Application Navigation with Activities and Intents 115
Working with Services 116
Receiving and Broadcasting Intents 117
Summary 117
References and More Information 118

6 Defining Your Application Using the Android Manifest File 119
Configuring Android Applications Using the Android Manifest File 119
Editing the Android Manifest File 120
Managing Your Application’s Identity 124
Versioning Your Application 125
Setting the Application Name and Icon 125
Enforcing Application System Requirements 125
Targeting Specific SDK Versions 126
Enforcing Application Platform Requirements 129
Working with External Libraries 130
Other Application Configuration Settings and Filters 131
Registering Activities in the Android Manifest 131
Designating a Primary Entry Point Activity for Your Application Using an Intent Filter 132

Configuring Other Intent Filters 132
Registering Other Application Components 133
Working with Permissions 133
Registering Permissions Your Application Requires 133
Registering Permissions Your Application Enforces 134
Exploring Other Manifest File Settings 135
Summary 136
References and More Information 136

7 Managing Application Resources 137
What Are Resources? 137
Storing Application Resources 137
Resource Value Types 138
Accessing Resources Programmatically 142
Setting Simple Resource Values Using Eclipse 143
Working with Different Types of Resources 146
Working with String Resources 146
Using String Resources as Format Strings 147
Working with String Arrays 149
Working with Boolean Resources 149
Working with Integer Resources 150
Working with Colors 151
Working with Dimensions 152
Working with Simple Drawables 153
Working with Images 154
Working with Animation 156
Working with Menus 158
Working with XML Files 159
Working with Raw Files 160
References to Resources 161
Working with Layouts 162
Referencing System Resources 167
Summary 168
References and More Information 168

III: Android User Interface Design Essentials

8 Exploring User Interface Screen Elements 171

Introducing Android Views and Layouts 171

Introducing the Android View 171

Introducing the Android Controls 171

Introducing the Android Layout 172

Displaying Text to Users with TextView 173

Configuring Layout and Sizing 173

Creating Contextual Links in Text 174

Retrieving Data from Users with EditText 176

Retrieving Text Input Using EditText Controls 176

Constraining User Input with Input Filters 178

Helping the User with Autocompletion 179

Giving Users Choices Using Spinner Controls 181

Allowing Simple User Selections with Buttons, Check Boxes, Switches, and Radio Groups 183

Using Basic Buttons 184

Using CheckBox and ToggleButton Controls 186

Using RadioGroup and RadioButton 187

Retrieving Dates and Times from Users 190

Using Indicators to Display Data to Users 191

Indicating Progress with ProgressBar 192

Adjusting Progress with SeekBar 194

Displaying Rating Data with RatingBar 194

Showing Time Passage with the Chronometer 195

Displaying the Time 196

Summary 197

References and More Information 198

9 Designing User Interfaces with Layouts 199

Creating User Interfaces in Android 199

Creating Layouts Using XML Resources 199

Creating Layouts Programmatically 201

Organizing Your User Interface 203

Using ViewGroup Subclasses for Layout Design 204

Using ViewGroup Subclasses as View Containers 204

Using Built-in Layout Classes 205
- Using FrameLayout 207
- Using LinearLayout 209
- Using RelativeLayout 211
- Using TableLayout 214
- Using GridLayout 216
- Using Multiple Layouts on a Screen 220

Using Container Control Classes 220
- Using Data-Driven Containers 221
- Organizing Screens with Tabs 226
- Adding Scrolling Support 229
- Exploring Other View Containers 230

Summary 231
- References and More Information 231

10 Working with Fragments 233

Understanding Fragments 233
- Understanding the Fragment Lifecycle 234
- Working with Special Types of Fragments 237
- Designing Fragment-Based Applications 238

Using the Android Support Package 247
- Adding Fragment Support to Legacy Applications 247
- Using Fragments in New Applications Targeting Older Platforms 248
- Linking the Android Support Package to Your Project 248

Summary 249
- References and More Information 250

11 Working with Dialogs 251

Choosing Your Dialog Implementation 251

Exploring the Different Types of Dialogs 252

Working with Dialogs: The Legacy Method 253
- Tracing the Lifecycle of a Dialog 254
- Working with Custom Dialogs 256

Working with Dialogs: The Fragment Method 257

Summary 260
- References and More Information 260

IV: Android Application Design Essentials

12 Using Android Preferences 263

Working with Application Preferences 263

Determining When Preferences Are Appropriate 263

Storing Different Types of Preference Values 264

Creating Private Preferences for Use by a Single Activity 264

Creating Shared Preferences for Use by Multiple Activities 265

Searching and Reading Preferences 265

Adding, Updating, and Deleting Preferences 266

Reacting to Preference Changes 267

Finding Preferences Data on the Android File System 267

Creating Manageable User Preferences 268

Creating a Preference Resource File 269

Using the PreferenceActivity Class 270

Summary 273

References and More Information 273

13 Working with Files and Directories 275

Working with Application Data on the Device 275

Practicing Good File Management 276

Understanding Android File Permissions 277

Working with Files and Directories 277

Exploring with the Android Application Directories 278

Working with Other Directories and Files on the Android File System 282

Summary 284

References and More Information 284

14 Using Content Providers 285

Exploring Android's Content Providers 285

Using the MediaStore Content Provider 286

Using the CallLog Content Provider 288

Using the Browser Content Provider 289

Using the CalendarContract Content Provider 291
Using the UserDictionary Content Provider 291
Using the VoicemailContract Content Provider 291
Using the Settings Content Provider 292
Using the Contacts Content Providers 292
Modifying Content Providers Data 297
Adding Records 297
Updating Records 298
Deleting Records 298
Using Third-Party Content Providers 299
Summary 300
References and More Information 300

15 Designing Compatible Applications 301
Maximizing Application Compatibility 301
Designing User Interfaces for Compatibility 303
Working with Fragments 305
Leveraging the Android Support Package 305
Supporting Specific Screen Types 305
Working with Nine-Patch Stretchable Graphics 306
Using the Working Square Principle 306
Providing Alternative Application Resources 308
Understanding How Resources Are Resolved 308
Organizing Alternative Resources with Qualifiers 309
Providing Resources for Different Orientations 316
Using Alternative Resources Programmatically 316
Organizing Application Resources Efficiently 316
Targeting Tablets, TVs, and Other New Devices 318
Targeting Tablet Devices 318
Targeting Google TV Devices 319
Summary 321
References and More Information 321

V: Publishing and Distributing Android Applications

16 The Android Software Development Process 325
An Overview of the Mobile Development Process 325
Choosing a Software Methodology 326

Understanding the Dangers of Waterfall Approaches 326
Understanding the Value of Iteration 327
Gathering Application Requirements 327
Determining Project Requirements 327
Developing Use Cases for Mobile Applications 329
Incorporating Third-Party Requirements 330
Managing a Device Database 330
Assessing Project Risks 333
Identifying Target Devices 333
Acquiring Target Devices 335
Determining the Feasibility of Application Requirements 336
Understanding Quality Assurance Risks 336
Writing Essential Project Documentation 337
Developing Test Plans for Quality Assurance Purposes 338
Providing Documentation Required by Third Parties 338
Providing Documentation for Maintenance and Porting 338
Leveraging Configuration Management Systems 339
Choosing a Source Control System 339
Implementing an Application Version System That Works 339
Designing Mobile Applications 340
Understanding Mobile Device Limitations 340
Exploring Common Mobile Application Architectures 340
Designing for Extensibility and Maintenance 341
Designing for Application Interoperability 342
Developing Mobile Applications 342
Testing Mobile Applications 343
Deploying Mobile Applications 343
Determining Target Markets 344
Supporting and Maintaining Mobile Applications 344
Track and Address Crashes Reported by Users 345
Testing Firmware Upgrades 345

Maintaining Adequate Application Documentation 345

Managing Live Server Changes 345

Identifying Low-Risk Porting Opportunities 345

Summary 346

References and More Information 346

17 Designing and Developing Bulletproof Android Applications 347

Best Practices in Designing Bulletproof Mobile Applications 347

Meeting Mobile Users' Demands 348

Designing User Interfaces for Mobile Devices 348

Designing Stable and Responsive Mobile Applications 349

Designing Secure Mobile Applications 351

Designing Mobile Applications for Maximum Profit 351

Leveraging Third-Party Quality Standards 352

Designing Mobile Applications for Ease of Maintenance and Upgrades 353

Leveraging Android Tools for Application Design 354

Avoiding Silly Mistakes in Android Application Design 355

Best Practices in Developing Bulletproof Mobile Applications 355

Designing a Development Process That Works for Mobile Development 356

Testing the Feasibility of Your Application Early and Often 356

Using Coding Standards, Reviews, and Unit Tests to Improve Code Quality 357

Handling Defects Occurring on a Single Device 359

Leveraging Android Tools for Development 360

Avoiding Silly Mistakes in Android Application Development 360

Summary 361

References and More Information 361

18 Testing Android Applications 363
Best Practices in Testing Mobile Applications 363
Designing a Mobile Application Defect Tracking System 363
Managing the Testing Environment 365
Maximizing Testing Coverage 367
Leveraging Android Tools for Android Application Testing 374
Avoiding Silly Mistakes in Android Application Testing 375
Summary 376
References and More Information 376

19 Publishing Your Android Application 377
Choosing the Right Distribution Model 377
Protecting Your Intellectual Property 378
Billing the User 379
Packaging Your Application for Publication 380
Preparing Your Code for Packaging 380
Packing and Signing Your Application 382
Testing the Release Version of Your Application Package 384
Distributing Your Application 385
Publishing on the Android Market 385
Signing Up for a Developer Account on the Android Market 385
Uploading Your Application to the Android Market 387
Uploading Application Marketing Assets 388
Configuring Application Listing Details 388
Configuring Application Publishing Options 390
Configuring Application Contact and Consent Information 390
Publishing Your Application on the Android Market 392
Managing Your Application on the Android Market 392
Publishing Using Other Alternatives 393
Self-Publishing Your Application 394

Summary 395
References and More Information 395

VI: Appendixes

A The Android Emulator Quick-Start Guide 399
Simulating Reality: The Emulator's Purpose 399
Working with Android Virtual Devices (AVDs) 401
Using the Android Virtual Device Manager 402
Creating an AVD 403
Launching the Emulator with a Specific AVD 407
Maintaining Emulator Performance 407
Configuring Emulator Startup Options 408
Launching an Emulator to Run an Application 408
Launching an Emulator from the Android Virtual Device Manager 410
Configuring the GPS Location of the Emulator 411
Calling Between Two Emulator Instances 413
Messaging between Two Emulator Instances 415
Interacting with the Emulator through the Console 416
Using the Console to Simulate Incoming Calls 416
Using the Console to Simulate SMS Messages 416
Using the Console to Send GPS Coordinates 418
Using the Console to Monitor Network Status 418
Using the Console to Manipulate Power Settings 418
Using Other Console Commands 419
Enjoying the Emulator 419
Understanding Emulator Limitations 420
References and More Information 421

B The Android DDMS Quick-Start Guide 423
Using DDMS with Eclipse and as a Standalone Application 423
Getting Up to Speed Using Key Features of DDMS 424
Working with Processes, Threads, and the Heap 425
Attaching a Debugger to an Android Application 425
Stopping a Process 426

Monitoring Thread Activity of an Android Application 426
Monitoring Heap Activity 427
Prompting Garbage Collection 428
Creating and Using an HPROF File 429
Using the Allocation Tracker 430
Working with the File Explorer 430
Browsing the File System of an Emulator or Device 432
Copying Files from the Emulator or Device 432
Copying Files to the Emulator or Device 433
Deleting Files on the Emulator or Device 433
Working with the Emulator Control 434
Simulating Incoming Voice Calls 434
Simulating Incoming SMS Messages 434
Sending a Location Fix 435
Taking Screen Captures of the Emulator and Device Screens 435
Working with Application Logging 436

C Eclipse IDE Tips and Tricks 439
Organizing Your Eclipse Workspace 439
Integrating with Source Control Services 439
Repositioning Tabs within Perspectives 440
Maximizing Windows 440
Minimizing Windows 440
Viewing Windows, Side by Side 440
Viewing Two Sections of the Same File 441
Closing Unwanted Tabs 441
Keeping Windows Under Control 441
Creating Custom Log Filters 441
Searching Your Project 442
Organizing Eclipse Tasks 442
Writing Code in Java 443
Using Autocomplete 443
Creating New Classes and Methods 443
Organizing Imports 443

Formatting Code 444
Renaming Almost Anything 444
Refactoring Code 444
Reorganizing Code 446
Using QuickFix 446
Providing Javadoc-Style Documentation 446
Resolving Mysterious Build Errors 447

Index 449

Acknowledgments

This book would never have been written without the guidance and encouragement we received from a number of supportive individuals, including our editorial team, coworkers, friends, and family. We'd like to thank the Android developer community, Google, and the Open Handset Alliance for their vision and expertise. Throughout this project, our editorial team at Pearson Education (Addison-Wesley) always had the right mix of professionalism and encouragement. Thanks especially to Trina MacDonald, Olivia Basegio, Songlin Qiu, and our crack team of technical reviewers: Doug Jones, Mike Wallace, and Mark Gjoel, (as well as Dan Galpin, Tony Hillerson, Ronan Schwarz, and Charles Stearns, who reviewed previous editions). Dan Galpin also graciously provided the clever Android graphics used for Tips, Notes, and Warnings. We'd also like to thank Ray Rischpater for his longtime encouragement and advice on technical writing. Amy Badger must be commended for her wonderful waterfall illustration, and we also thank Hans Bodlaender for letting us use the nifty chess font he developed as a hobby project.

About the Authors

Lauren Darcey is responsible for the technical leadership and direction of a small software company specializing in mobile technologies, including Android, 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.

She spends her copious free time traveling the world with her geeky mobile-minded husband and is an avid nature photographer. Her 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, 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 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 is catching on that he should be writing his own bio.

Introduction

Pioneered by the Open Handset Alliance and Google, Android is a popular, free, open-source mobile platform that has taken the wireless world by storm. This book, and the next volume, *Android Wireless Application Development Volume II: Advanced Topics,* provide comprehensive guidance for software development teams on designing, developing, testing, debugging, and distributing professional Android applications. If you're a veteran mobile developer, you can find tips and tricks to streamline the development process and take advantage of Android's unique features. If you're new to mobile development, these books provide everything you need to make a smooth transition from traditional software development to mobile development—specifically, its most promising platform: Android.

Who Should Read This Book

This book include tips for successful mobile development based upon our years in the mobile industry and covers everything you need to know in order to run a successful Android project from concept to completion. We cover how the mobile software process differs from traditional software development, including tricks to save valuable time and pitfalls to avoid. Regardless of the size of your project, this book is for you.

This book was written for several audiences:

- **Software developers who want to learn to develop professional Android applications.** The bulk of this book is targeted at software developers with Java experience who do not necessarily have mobile development experience. More seasoned developers of mobile applications can learn how to take advantage of Android and how it differs from the other technologies of the mobile development market today.
- **Quality assurance personnel tasked with testing Android applications.** Whether they are black-box or white-box testing, quality assurance engineers can find this book invaluable. We devote several chapters to mobile QA concerns, including topics such as developing solid test plans and defect-tracking systems for mobile applications, how to manage handsets, and how to test applications thoroughly using all the Android tools available.
- **Project managers planning and managing Android development teams.** Managers can use this book to help plan, hire, and execute Android projects from start to finish. We cover project risk management and how to keep Android projects running smoothly.

- **Other audiences.** This book is useful not only to the software developer, but also to the corporation looking at potential vertical market applications, the entrepreneur thinking about a cool phone application, and the hobbyist looking for some fun with his or her new phone. Businesses seeking to evaluate Android for their specific needs (including feasibility analysis) can also find the information provided valuable. Anyone with an Android handset and a good idea for a mobile application can put the information provided in this book to use for fun and profit.

Key Questions Answered in This Volume

This volume of the book answers the following questions:

1. What is Android? How do the SDK versions differ?
2. How is Android different from other mobile technologies, and how can developers take advantage of these differences?
3. How do developers use the Eclipse Development Environment for Java to develop and debug Android applications on the emulator and handsets?
4. How are Android applications structured?
5. How do developers design robust user interfaces for mobile—specifically, for Android?
6. What capabilities does the Android SDK have and how can developers use them?
7. How does the mobile development process differ from traditional desktop development?
8. What development strategies work best for Android development?
9. What do managers, developers, and testers need to look for when planning, developing, and testing a mobile development application?
10. How do mobile teams design bulletproof Android applications for publication?
11. How do mobile teams package Android applications for deployment?
12. How do mobile teams make money from Android applications?
13. And, finally, what is new in this edition of the book?

How These Books Are Structured

We wrote the first edition of this book before the Android SDK was released. Now, three years and 14 Android SDK releases later, there is so much to talk about that we've had to divide the content of *Android Wireless Application Development* into two separate volumes for this, the third edition.

Android Wireless Application Development Volume I: Android Essentials focuses on Android essentials, including setting up your development environment, understanding the application lifecycle, user interface design, developing for different types of devices, and the mobile software process from design and development to testing and publication of commercial-grade applications.

Android Wireless Application Development Volume II: Advanced Topics focuses on advanced Android topics, including leveraging various Android APIs for threading, networking, location-based services, hardware sensors, animation, graphics, and more. Coverage of advanced Android application components, such as services, application databases, content providers, and intents, is also included. Developers learn to design advanced user interface components and integrate their applications deeply into the platform. Finally, developers learn how to extend their applications beyond traditional boundaries using optional features of the Android platform, including the Android Native Development Kit (NDK), Cloud-To-Device Messaging service (C2DM), Android Market In-Application Billing APIs, Google Analytics APIs, and more.

Android Wireless Application Development Volume I: Android Essentials is divided into six parts. The first four parts are primarily of interest to developers; Part V provides lots of helpful information for project managers and quality assurance personnel as well as developers. Part VI includes several helpful appendixes to help you get up and running with the most important Android tools.

Here is an overview of the various parts in this book:

- **Part I: An Overview of the Android Platform**
  Part I provides an introduction to Android, explaining how it differs from other mobile platforms. You become familiar with the Android SDK and tools, install the development tools, and write and run your first Android application—on the emulator and on a handset.
- **Part II: Android Application Basics**
  Part II introduces the design principles necessary to write Android applications. You learn how Android applications are structured and how to include resources, such as strings, graphics, and user interface components, in your projects.
- **Part III: Android User Interface Design Essentials**
  Part III dives deeper into how user interfaces are designed in Android. You learn about the core user interface element in Android: the `View`. You also learn about the most common user interface controls and layouts provided in the Android SDK.
- **Part IV: Android Application Design Essentials**
  Part IV covers the features used by most Android applications, including storing persistent application data using preferences and working with files, directories, and content providers. You also learn how to design applications that will run smoothly on many different Android devices.

- **Part V: Publishing and Distributing Android Applications**
  Part V covers the software development process for mobile, from start to finish, with tips and tricks for project management, software developers, and quality assurance personnel.
- **Part VI: Appendixes**
  Part VI includes two helpful quick-start guides for the Android development tools—the emulator and DDMS—as well as an appendix of Eclipse tips and tricks.

An Overview of Changes in This Edition

When we began writing the first edition of this book, there were no Android devices on the market. One Android device became available shortly after we started writing, and it was available only in the United States. Today there are hundreds of devices shipping all over the world—smartphones, tablets, e-book readers, and specialty devices such as the Google TV. The Android platform has gone through extensive changes since the first edition of this book was published. The Android SDK has many new features and the development tools have received many much-needed upgrades. Android, as a technology, is now on solid footing within the mobile marketplace.

In this new edition, we took the opportunity to do a serious overhaul of the book content. But don't worry, it's still the book readers loved the first (and second!) time; it's just bigger, better, and more comprehensive. In order to cover more of the exciting topics available to Android developers, we had to divide the book into two volumes. In addition to adding tons of new content, we've retested and upgraded all existing content (text and sample code) for use with the latest Android SDKs available while still remaining backward compatible. The Android development community is diverse, and we aim to support all developers, regardless of which devices they are developing for. This includes developers who need to target nearly all platforms, so coverage in some key areas of older SDKs continues to be included because it's often the most reasonable option for compatibility.

Here are some of the highlights of the additions and enhancements we've made to this edition:

- Coverage of the latest and greatest Android tools and utilities.
- Updates to all existing chapters, often with some entirely new sections.
- New chapters, which cover new SDK features or expand upon those covered in previous editions.
- Updated sample code and applications, conveniently organized by chapter.
- Topics such as Android manifest files, content providers, designing apps, and testing now have their own chapters.

- Coverage of hot topics such as application compatibility, designing for different devices, and working with relatively new user interface components such as fragments.
- Even more tips and tricks from the trenches to help you design, develop, and test applications for different device targets, including an all-new chapter on tackling compatibility issues.

As you can see, we cover many of the hottest and most exciting features that Android has to offer. We didn't take this review lightly; we touched every existing chapter, updated content, and added many new chapters as well. Finally, we included many additions, clarifications, and, yes, even a few fixes based on the feedback from our fantastic (and meticulous) readers. Thank you!

Development Environment Used in This Book

The Android 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) and Version 3.6 (Helios)
- Eclipse JDT plug-in and Web Tools Platform (WTP)
- Java SE Development Kit (JDK) 6 Update 26
- Android SDK Version 2.3.4, API Level 10 (Gingerbread MR1); Android SDK Version 3.2, API Level 13 (Honeycomb MR2); Android SDK Version 4.0, API Level 14 (Ice Cream Sandwich)
    1. ADT Plug-in for Eclipse 15.0.0
    2. SDK Tools Revision 15

  Android Devices: Samsung Nexus S, HTC Evo 4G, Motorola Droid 3, Samsung Galaxy Tab 10.1, Motorola Xoom, Motorola Atrix 4G, and Sony Ericsson Xperia Play

The Android platform continues to aggressively grow in market share against competing mobile platforms, such as Apple iOS and BlackBerry. New and exciting types of devices reach consumers' hands at a furious pace, with new editions of the Android platform appearing all the time. Developers can no longer ignore Android as a target platform if they want to reach the smartphone (or smart-device) users of today and tomorrow.

Android's latest major platform update, Android 4.0—frequently called by its codename, Ice Cream Sandwich, or just ICS—merges the smartphone-centric Android 2.3.x (Gingerbread) and the tablet-centric Android 3.x (Honeycomb) platform editions into a single SDK for all smart-devices, be they phones, tablets, televisions, or toasters. This book features the latest SDK and tools available, but it does not focus on them to the detriment of popular legacy versions of the platform. This book is meant to be an overall reference to help developers support all popular devices on the market today. As of the writing of this book, only a very small percentage (less than 5%) of users' devices are running Android 3.0 or 4.0. Of course, some devices will receive upgrades, and users will purchase new Ice Cream Sandwich devices as they become available, but for now, developers need to straddle this gap and support numerous versions of Android to reach the majority of users in the field.

So what does this mean for this book? It means we provide both legacy API support and discuss some of the newer APIs only available in later versions of the Android SDK. We discuss strategies for supporting all (or at least most) users in terms of compatibility. And we provide screenshots that highlight different versions of the Android SDK, because each major revision has brought with it a change in the look and feel of the overall platform. That said, we are assuming that you are downloading the latest Android tools, so we provide screenshots and steps that support the latest tools available at the time of writing, not legacy tools. Those are the boundaries we set when trying to determine what to include or leave out of this book.

Supplementary Materials Available

The source code that accompanies this book is available for download on the publisher website: www.informit.com/title/9780321813831. The source code is also available for download from our book website: http://androidbook.blogspot.com/p/book-code-downloads.html (http://goo.gl/kyAsN). You'll also find a variety of Android topics discussed at our book website (http://androidbook.blogspot.com). For example, we present reader feedback, questions, and further information. You can find links to our various technical articles on our book website as well.

Where to Find More Information

There is a vibrant, helpful Android developer community on the Web. Here are a number of useful websites for Android developers and followers of the wireless industry:

- **Android Developer website:** The Android SDK and developer reference site:
  http://developer.android.com or http://d.android.com
- **Stack Overflow:** The Android website with great technical information (complete with tags) and an official support forum for developers:
  http://stackoverflow.com/questions/tagged/android
- **Open Handset Alliance:** Android manufacturers, operators, and developers:
  http://www.openhandsetalliance.com

- **Android Market:** Buy and sell Android applications:
  http://www.android.com/market
- **Mobiletuts+:** Mobile development tutorials, including Android:
  http://mobile.tutsplus.com/category/tutorials/android
- **anddev.org:** An Android developer forum:
  http://www.anddev.org
- **Google Team Android Apps:** Open-source Android applications:
  http://apps-for-android.googlecode.com
- **Android Tools Project Site:** The tools team discusses updates and changes:
  https://sites.google.com/a/android.com/tools/recent
- **FierceDeveloper:** A weekly newsletter for wireless developers:
  http://www.fiercedeveloper.com
- **Wireless Developer Network:** Daily news on the wireless industry:
  http://www.wirelessdevnet.com
- **XDA-Developers Android Forum**: From general development to ROMs:
  http://forum.xda-developers.com/forumdisplay.php?f=564
- **Developer.com:** A developer-oriented site with mobile articles:
  http://www.developer.com

Conventions Used in This Book

This book uses the following conventions:

- ➥ is used to signify to readers that the authors meant for the continued code to appear on the same line. No indenting should be done on the continued line.
- Code or programming terms are set in `monospace` text.
- Java import statements, exception handling, and error checking are often removed from printed code examples for clarity and to keep the book a reasonable length.

This book also presents information in the following sidebars:

Tip

Tips provide useful information or hints related to the current text.

Note

Notes provide additional information that might be interesting or relevant.

Warning

Warnings provide hints or tips about pitfalls that may be encountered and how to avoid them.

Contacting the Authors

We welcome your comments, questions, and feedback. We invite you to visit our blog at:

- http://androidbook.blogspot.com

Or email us at:

- androidwirelessdev+awad3ev1@gmail.com

Circle us on Google+:

- Lauren Darcey: http://goo.gl/P3RGo
- Shane Conder: http://goo.gl/BpVJh

3

Writing Your First Android Application

You should now have a workable Android development environment set up on your computer. Hopefully, you also have an Android device as well. Now it's time for you to start writing some Android code. In this chapter, you learn how to add and create Android projects in Eclipse and verify that your Android development environment is set up correctly. You also write and debug your first Android application in the software emulator and on an Android device.

Note

The Android development tools are updated frequently. We have made every attempt to provide the latest steps for the latest tools. However, these steps and the user interfaces described in this chapter may change at any time. Please review the Android development website (http://d.android.com/sdk) and our book website (http://androidbook.blogspot.com) for the latest information.

Testing Your Development Environment

The best way to make sure you configured your development environment correctly is to take an existing Android application and run it. You can do this easily by using one of the sample applications provided as part of the Android SDK in the `samples` subdirectory found where your Android SDK is installed.

Within the Android SDK sample applications, you will find a classic game called *Snake* (http://goo.gl/wRojX). To build and run the Snake application, you must create a new Android project in your Eclipse workspace based on the existing Android sample project, create an appropriate Android Virtual Device (AVD) profile, and configure a launch configuration for that project. After you have everything set up correctly, you can build the application and run it on the Android emulator and on an Android device. By testing your development environment with a sample application, you can rule out project configuration and coding issues and focus on determining whether the tools are set up properly for Android development. After this fact has been established, you can move on to writing and compiling your own applications.

Adding the Snake Project to Your Eclipse Workspace

The first thing you need to do is add the Snake project to your Eclipse workspace. To do this, follow these steps:

1. Choose File, New, Other….
2. Choose Android, Android Sample Project (see Figure 3.1). Click Next.

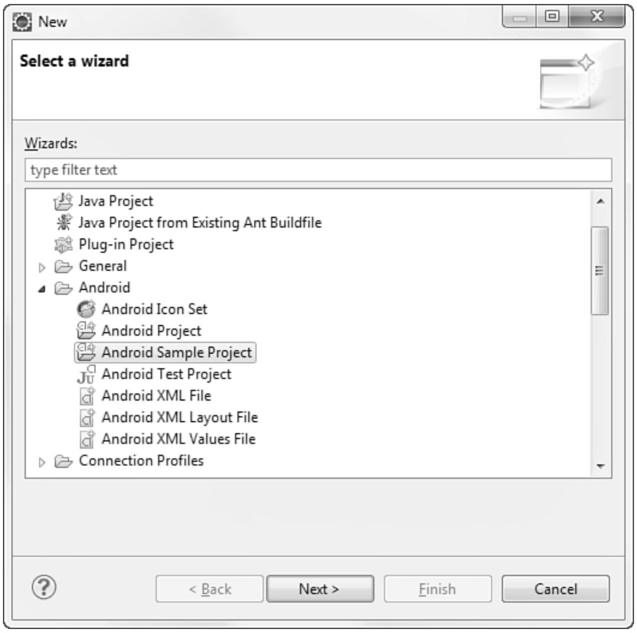
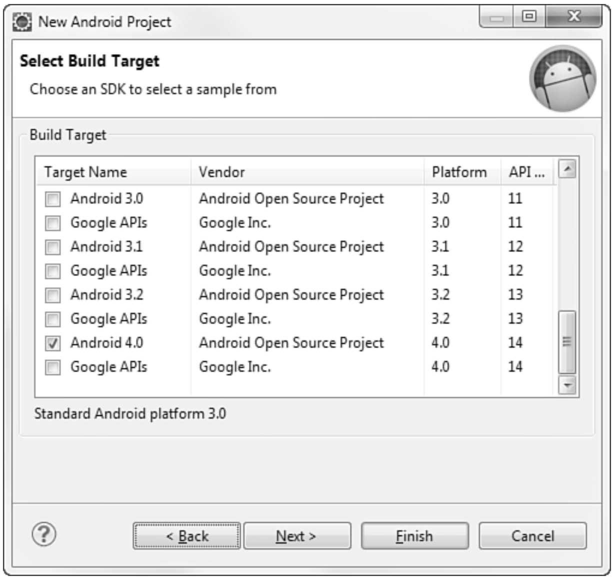


Figure 3.1 Creating a new Android project.

3. Choose your build target (see Figure 3.2). In this case, we've picked Android 4.0, API Level 14, from the Android Open Source Project. Click Next.
4. Next, select which sample you want to create (see Figure 3.3). Choose Snake.
5. Click Finish. You now see the Snake project files in your workspace (see Figure 3.4).


New Android Project

Select Build Target

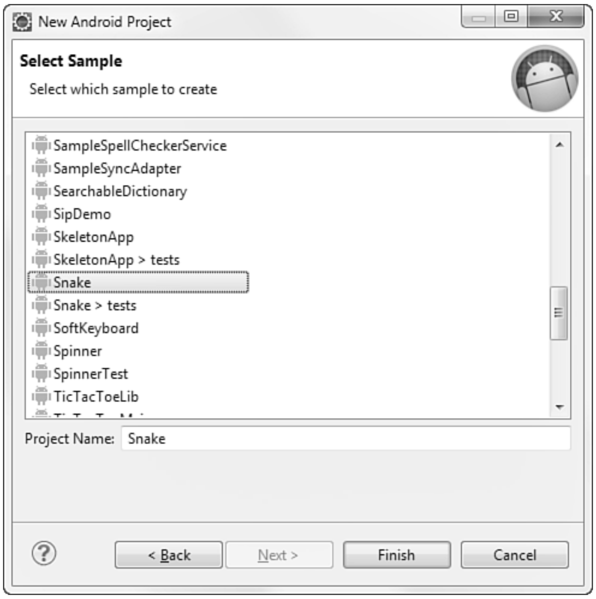
Choose an SDK to select a sample from

Build Target

Target Name	Vendor	Platform	API ...
☐ Android 3.0	Android Open Source Project	3.0	11
☐ Google APIs	Google Inc.	3.0	11
☐ Android 3.1	Android Open Source Project	3.1	12
☐ Google APIs	Google Inc.	3.1	12
☐ Android 3.2	Android Open Source Project	3.2	13
☐ Google APIs	Google Inc.	3.2	13
✓ Android 4.0	Android Open Source Project	4.0	14
☐ Google APIs	Google Inc.	4.0	14

Standard Android platform 3.0

Figure 3.2 Choose an API level for the sample.


New Android Project

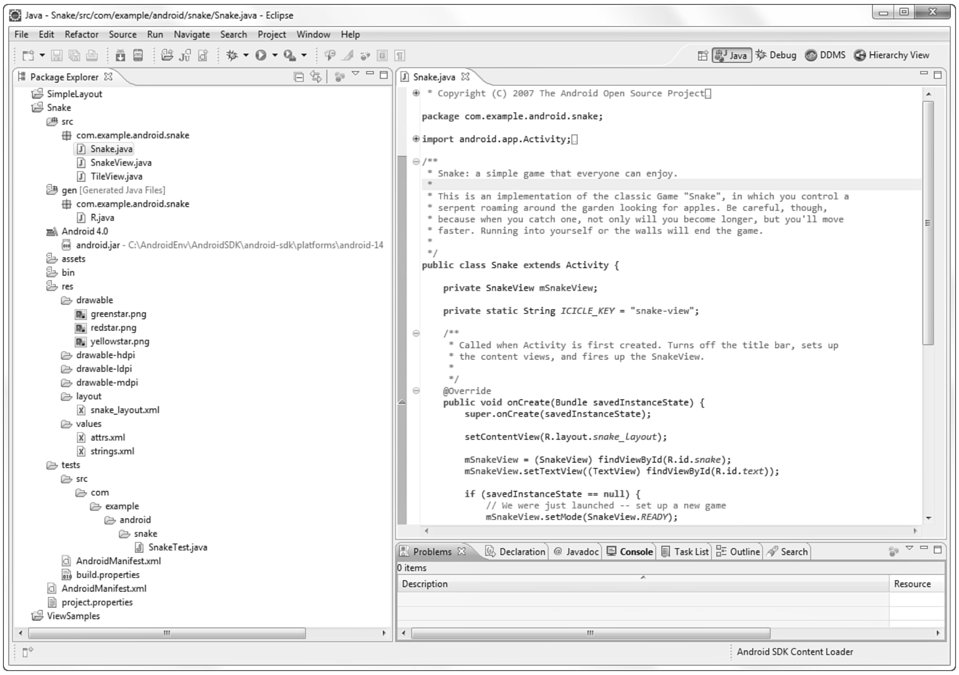
Select Sample

Select which sample to create

- SampleSpellCheckerService
- SampleSyncAdapter
- SearchableDictionary
- SipDemo
- SkeletonApp
- SkeletonApp > tests
- Snake
- Snake > tests
- SoftKeyboard
- Spinner
- SpinnerTest
- TicTacToeLib

Project Name: Snake

Figure 3.3 Picking the sample project.


Java - Snake/src/com/example/android/snake/Snake.java - Eclipse
File Edit Refactor Source Run Navigate Search Project Window Help
Java
Debug
DDMS
Hierarchy View
Package Explorer
Snake.java
Problems
Declaration
Javadoc
Console
Task List
Outline
Search
0 items
Description
Resource
Android SDK Content Loader

```
SimpleLayout
Snake
    src
        com.example.android.snake
            Snake.java
            SnakeView.java
            TileView.java
    gen [Generated Java Files]
        com.example.android.snake
            R.java
    Android 4.0
        android.jar - C:\AndroidEnv\AndroidSDK\android-sdk\platforms\android-14
    assets
    bin
    res
        drawable
            greenstar.png
            redstar.png
            yellowstar.png
        drawable-hdpi
        drawable-ldpi
        drawable-mdpi
        layout
            snake_layout.xml
        values
            attrs.xml
            strings.xml
    tests
        src
            com
                example
                    android
                        snake
                            SnakeTest.java
        AndroidManifest.xml
        build.properties
    AndroidManifest.xml
    project.properties
ViewSamples
```

```java
 * Copyright (C) 2007 The Android Open Source Project

package com.example.android.snake;

import android.app.Activity;

/**
 * Snake: a simple game that everyone can enjoy.
 *
 * This is an implementation of the classic Game "Snake", in which you control a
 * serpent roaming around the garden looking for apples. Be careful, though,
 * because when you catch one, not only will you become longer, but you'll move
 * faster. Running into yourself or the walls will end the game.
 *
 */
public class Snake extends Activity {

    private SnakeView mSnakeView;

    private static String ICICLE_KEY = "snake-view";

    /**
     * Called when Activity is first created. Turns off the title bar, sets up
     * the content views, and fires up the SnakeView.
     *
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.snake_layout);

        mSnakeView = (SnakeView) findViewById(R.id.snake);
        mSnakeView.setTextView((TextView) findViewById(R.id.text));

        if (savedInstanceState == null) {
            // We were just launched -- set up a new game
            mSnakeView.setMode(SnakeView.READY);
```

Figure 3.4 The Snake project files.

Warning

Occasionally Eclipse shows the error "Project 'Snake' is missing required source folder: gen" when you're adding an existing project to the workspace. If this happens, navigate to the project file called `R.java` under the `/gen` directory and delete it. The `R.java` file is automatically regenerated and the error should disappear. Performing a Clean operation followed by a Build operation does not always solve this problem.

Creating an Android Virtual Device (AVD) for Your Snake Project

The next step is to create an AVD that describes what type of device you want to emulate when running the Snake application. This AVD profile describes what type of device you want the emulator to simulate, including which Android platform to support. You do not need to create new AVDs for each application, only for each device you want to emulate. You can specify different screen sizes and orientations, and you can specify whether the emulator has an SD card and, if it does, what capacity the card is.

For the purposes of this example, an AVD for the default installation of Android 2.3.3 suffices. Here are the steps to create a basic AVD:

1. Launch the Android Virtual Device Manager from within Eclipse by clicking the little Android device icon on the toolbar (📱). If you cannot find the icon, you can also launch the manager through the Window menu of Eclipse.
2. Click the New button.
3. Choose a name for your AVD. Because we are going to take all the defaults, give this AVD a name of `Android_Vanilla4.0`.
4. Choose a build target. We want a typical Android 4.0 device, so choose Google APIs (Google Inc.) – API Level 14 from the drop-down menu. This will include the Google Android applications, such as the Maps application, as part of the platform image.
5. Choose an SD card capacity, in either kibibytes or mibibytes. Not familiar with kibibytes? See this Wikipedia entry: http://goo.gl/N3Rdd. This SD card image will take up space on your hard drive, so choose something reasonable, such as 1024MiB.
6. Seriously consider enabling the Snapshot feature. This feature greatly improves emulator startup performance. See Appendix A, "The Android Emulator Quick-Start Guide," for details.

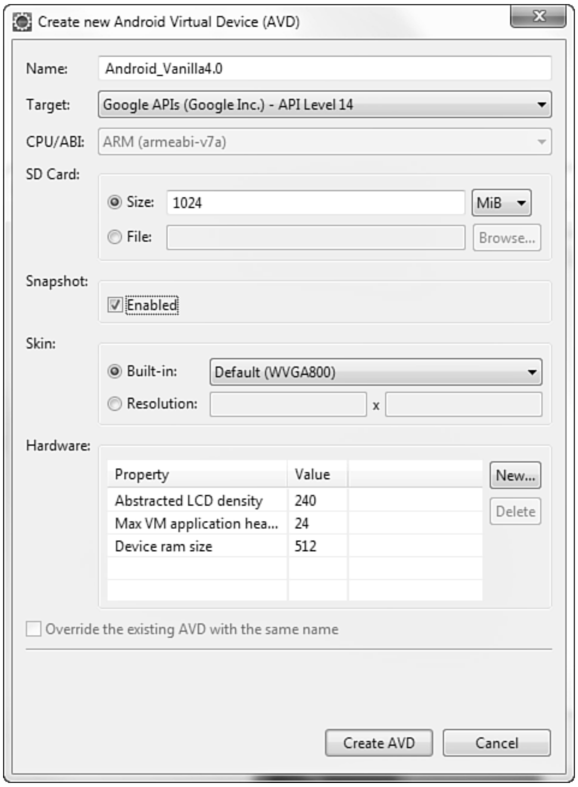
Warning

As of this writing, there's a known issue with the Android Tools R14 emulator that prevents the snapshot feature from working correctly. See http://goo.gl/pnMt0 for more information on the current known issues.

7. Choose a skin. This option controls the different resolutions of the emulator. In this case, we use the recommended WVGA800 screen. (The emulator in Tools R14 has known performance issues with running the standard Android 4.0 screen resolution of WXGA720.) This skin most directly correlates to the popular 4.0 devices. Feel free to choose the most appropriate skin to match the Android device on which you plan to run the application.

   Your project settings will look like Figure 3.5.
8. Click the Create AVD button and then wait for the operation to complete.
9. Click Finish. Because the AVD manager formats the memory allocated for SD card images, creating AVDs with SD cards could take a few moments.

For more information on creating different types of AVDs, check out Appendix A.


Create new Android Virtual Device (AVD)

Name: Android_Vanilla4.0
Target: Google APIs (Google Inc.) - API Level 14
CPU/ABI: ARM (armeabi-v7a)
SD Card:
(●) Size: 1024 MiB
( ) File: Browse...
Snapshot: [✓] Enabled
Skin:
(●) Built-in: Default (WVGA800)
( ) Resolution: x
Hardware:

Property	Value
Abstracted LCD density	240
Max VM application hea...	24
Device ram size	512

[ ] Override the existing AVD with the same name

Figure 3.5 Creating a new AVD.

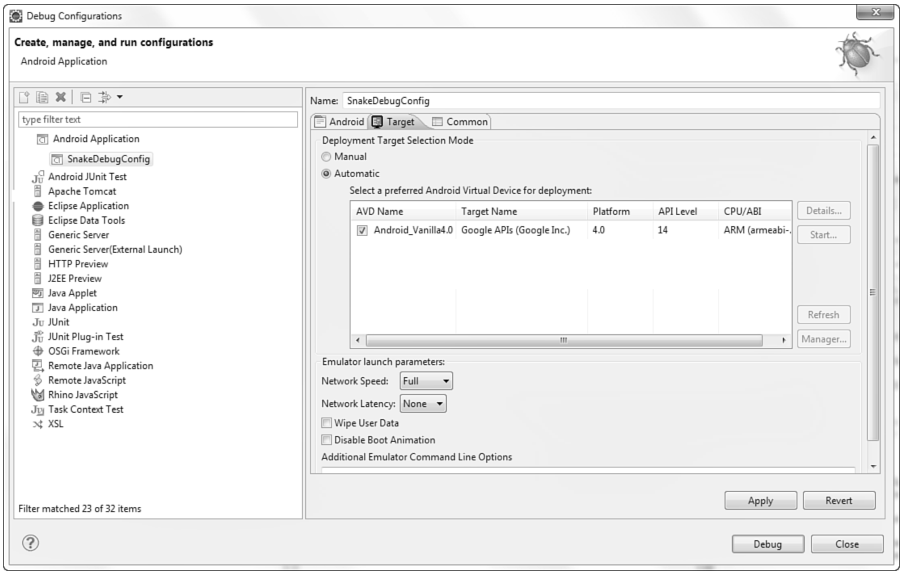
Creating a Launch Configuration for Your Snake Project

Next, you must create a launch configuration in Eclipse to configure under what circumstances the Snake application builds and launches. The launch configuration is where you configure the emulator options to use and the entry point for your application.

You can create Run configurations and Debug configurations separately, each with different options. These configurations are created under the Run menu in Eclipse (Run, Run Configurations and Run, Debug Configurations). Follow these steps to create a basic Debug configuration for the Snake application:

1. Choose Run, Debug Configurations.
2. Double-click Android Application to create a new configuration.
3. Name your Debug configuration `SnakeDebugConfiguration`.
4. Choose the project by clicking the Browse button and choosing the Snake project.

5. Switch to the Target tab and, from the preferred AVD list, choose the `Android_Vanilla4.0` AVD created earlier, as shown in Figure 3.6.


Debug Configurations

Create, manage, and run configurations
Android Application

type filter text
- Android Application
  - SnakeDebugConfig
- Android JUnit Test
- Apache Tomcat
- Eclipse Application
- Eclipse Data Tools
- Generic Server
- Generic Server(External Launch)
- HTTP Preview
- J2EE Preview
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- OSGi Framework
- Remote Java Application
- Remote JavaScript
- Rhino JavaScript
- Task Context Test
- XSL

Filter matched 23 of 32 items

Name: SnakeDebugConfig

Android | Target | Common

Deployment Target Selection Mode
( ) Manual
(•) Automatic
Select a preferred Android Virtual Device for deployment:

AVD Name	Target Name	Platform	API Level	CPU/ABI
☑ Android_Vanilla4.0	Google APIs (Google Inc.)	4.0	14	ARM (armeabi-)

Details... Start... Refresh Manager...

Emulator launch parameters:
Network Speed: Full
Network Latency: None
☐ Wipe User Data
☐ Disable Boot Animation
Additional Emulator Command Line Options

Apply Revert
Debug Close

Figure 3.6 The Snake application Debug configuration in Eclipse.

You can set other emulator and launch options on the Target and Common tabs, but for now we are leaving the defaults as they are.

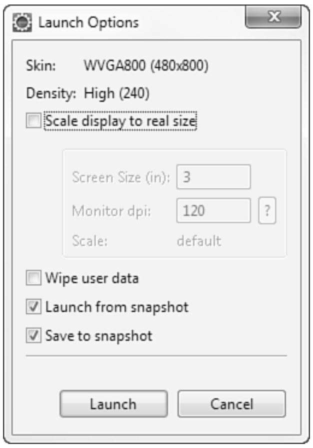
Running the Snake Application in the Android Emulator

Now you can run the Snake application using the following steps:

1. Choose the Debug As icon drop-down menu on the toolbar ( ).
2. Pull the drop-down menu and choose the `SnakeDebugConfiguration` you created. If you do not see your new configuration listed, find it in the Debug Configurations listing and click the Debug button. Subsequent launches can be initiated from the little bug drop-down.
3. The Android emulator starts up; this might take a few moments to initialize. Then the application will be installed or reinstalled onto the emulator.

Tip

It can take a long time for the emulator to start up, even on very fast computers. You might want to leave it around while you work and reattach to it as needed. The tools in Eclipse handle reinstalling the application and re-launching the application, so you can more easily keep the emulator loaded all the time. This is another reason to enable the Snapshot feature for each AVD. You can also use the Start button on the Android Virtual Device Manager to load up an emulator before you need it. Launching the AVD this way also gives you some additional options such as screen scaling (see Figure 3.7), which can be used to either fit the AVD on your screen if it's very high resolution or more closely emulate the size it might be on real hardware.


Launch Options

Skin: WVGA800 (480x800)

Density: High (240)

☐ Scale display to real size

Screen Size (in): 3

Monitor dpi: 120

Scale: default

☐ Wipe user data

☑ Launch from snapshot

☑ Save to snapshot

Figure 3.7 Configuring AVD launch options.

4. If necessary, swipe the screen from left to right to unlock the emulator, as shown in Figure 3.8.
5. The Snake application starts and you can play the game, as shown in Figure 3.9.

You can interact with the Snake application through the emulator and play the game. You can also launch the Snake application from the Application drawer at any time by clicking its application icon. There is no need to shut down and restart the emulator every time you rebuild and reinstall your application for testing. Simply leave the emulator running on your computer in the background while you work in Eclipse and then redeploy using the Debug configuration again.

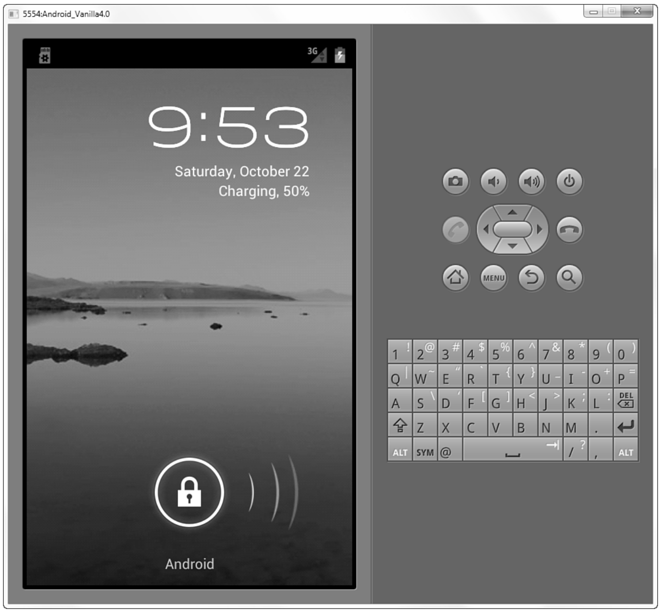


Figure 3.8 The Android emulator launching (locked).

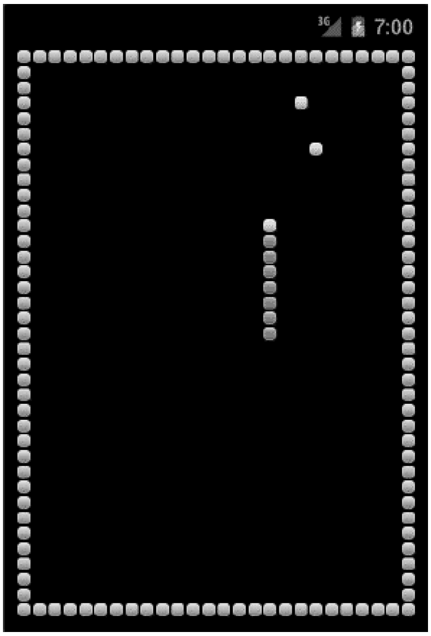


Figure 3.9 The Snake game in the Android emulator.

Building Your First Android Application

Now it's time to write your first Android application from scratch. To get your feet wet, you will start with a simple "Hello World" application and build upon it to explore some of the features of the Android platform in more detail.

Tip

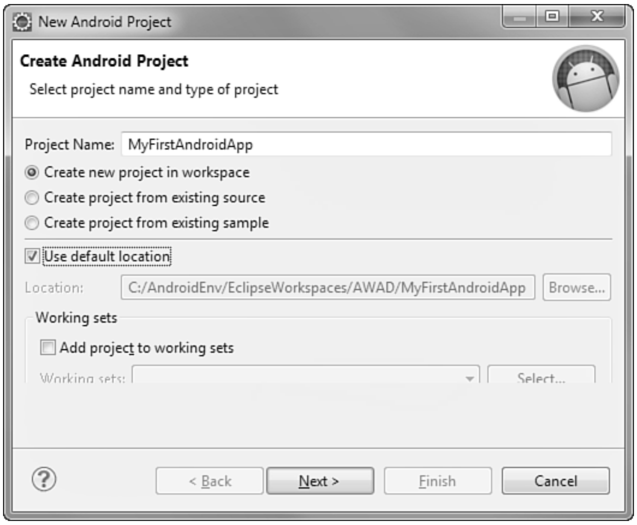
The code examples provided in this chapter are taken from the MyFirstAndroidApp application. The source code for the MyFirstAndroidApp application is provided for download on the book's websites.

Creating and Configuring a New Android Project

You can create a new Android application in much the same way as when you added the Snake application to your Eclipse workspace.

The first thing you need to do is create a new project in your Eclipse workspace. The Android Project Wizard creates all the required files for an Android application. Follow these steps within Eclipse to create a new project:

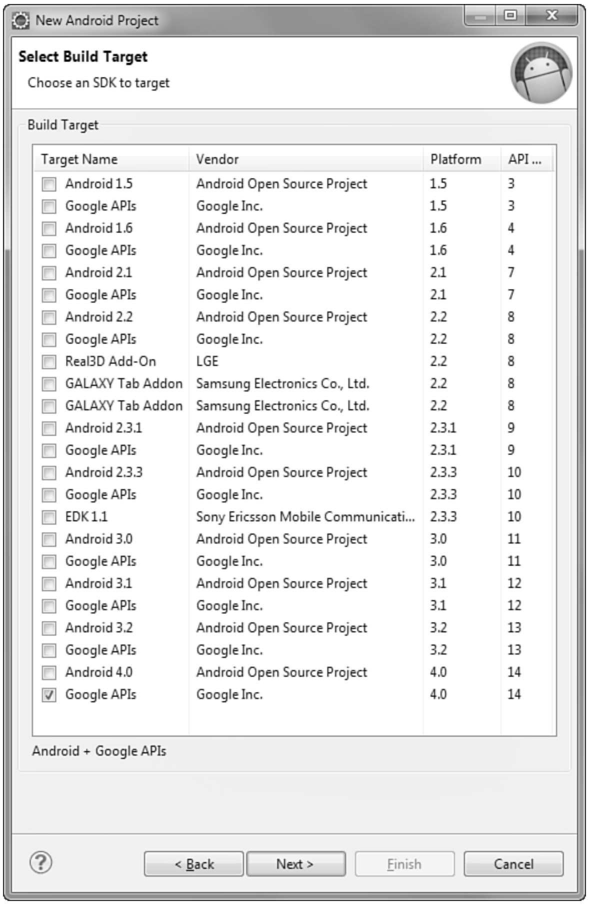
1. Choose File, New, Android Project, or choose the Android Project creator icon, which looks like a folder ( ), on the Eclipse toolbar.
2. Choose a project name, as shown in Figure 3.10. In this case, name the project `MyFirstAndroidApp`.



Project Name: MyFirstAndroidApp
(●) Create new project in workspace
( ) Create project from existing source
( ) Create project from existing sample
[✓] Use default location
Location: C:/AndroidEnv/EclipseWorkspaces/AWAD/MyFirstAndroidApp
Working sets
[ ] Add project to working sets

Figure 3.10 Configuring a new Android project.

3. Choose a location for the project files. Because this is a new project, select the Create New Project in Workspace radio button. Check the Use Default Location check box or change the directory to wherever you want to store the source files. Click Next.
4. Select a build target for your application, as shown in Figure 3.11. Choose a target that is compatible with the Android devices you have in your possession. For this example, you might use the Android 2.3.3 target or, for Ice Cream Sandwich devices, Android 4.0 (API Level 14). Click Next.



Select Build Target

Choose an SDK to target

Build Target

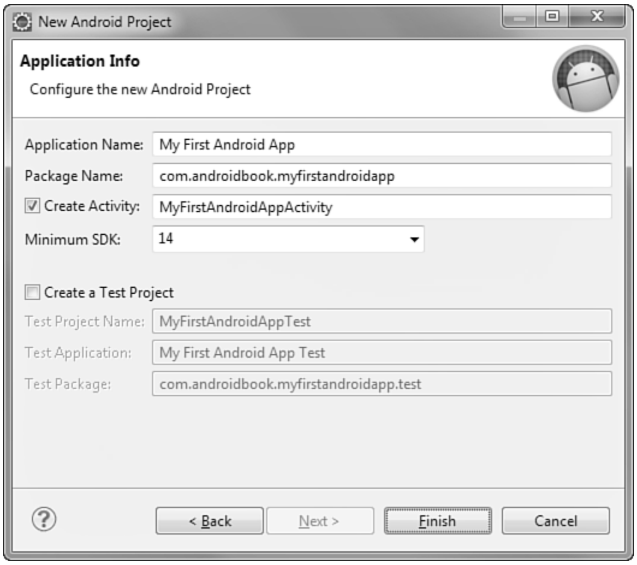
Target Name	Vendor	Platform	API ...
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Google APIs	Google Inc.	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☐ Google APIs	Google Inc.	1.6	4
☐ Android 2.1	Android Open Source Project	2.1	7
☐ Google APIs	Google Inc.	2.1	7
☐ Android 2.2	Android Open Source Project	2.2	8
☐ Google APIs	Google Inc.	2.2	8
☐ Real3D Add-On	LGE	2.2	8
☐ GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
☐ GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
☐ Android 2.3.1	Android Open Source Project	2.3.1	9
☐ Google APIs	Google Inc.	2.3.1	9
☐ Android 2.3.3	Android Open Source Project	2.3.3	10
☐ Google APIs	Google Inc.	2.3.3	10
☐ EDK 1.1	Sony Ericsson Mobile Communicati...	2.3.3	10
☐ Android 3.0	Android Open Source Project	3.0	11
☐ Google APIs	Google Inc.	3.0	11
☐ Android 3.1	Android Open Source Project	3.1	12
☐ Google APIs	Google Inc.	3.1	12
☐ Android 3.2	Android Open Source Project	3.2	13
☐ Google APIs	Google Inc.	3.2	13
☐ Android 4.0	Android Open Source Project	4.0	14
✓ Google APIs	Google Inc.	4.0	14

Android + Google APIs

Figure 3.11 Choosing a build target for a new Android project.

5. Configure your application information. Choose an application name. The application name is the "friendly" name of the application and the name shown with the icon on the application launcher. In this case, the application name is "My First Android App."
6. Choose a package name. Here you should follow standard package namespace conventions for Java. Because all our code examples in this book fall under the `com.androidbook.*` namespace, we will use the package name `com.androidbook.myfirstandroidapp`, but you are free to choose your own package name.
7. Check the Create Activity check box. This instructs the wizard to create a default launch activity for the application. Call this `Activity` class `MyFirstAndroidAppActivity`.
8. Set the minimum SDK version. This value should be the same or lower than the target SDK API level. Because our application will be compatible with just about any Android device, you can set this number low (like to 4 to represent Android 1.6) or at the target API level to avoid annoying warnings in Eclipse. Make sure you set the minimum SDK version to encompass any test devices you have available so you can successfully install the application on them.

   Your project settings should look like Figure 3.12.
9. Finally, click the Finish button.



Application Name: My First Android App
Package Name: com.androidbook.myfirstandroidapp
[✓] Create Activity: MyFirstAndroidAppActivity
Minimum SDK: 14
[ ] Create a Test Project
Test Project Name: MyFirstAndroidAppTest
Test Application: My First Android App Test
Test Package: com.androidbook.myfirstandroidapp.test

Figure 3.12 Configuring My First Android App using the Android Project Wizard.

Core Files and Directories of the Android Application

Every Android application has a set of core files that are created and used to define the functionality of the application. The following files are created by default with a new Android application:

- **`AndroidManifest.xml`**—The central configuration file for the application. It defines your application's capabilities and permissions as well as how it runs.
- **`project.properties`**—A generated build file used by Eclipse and the Android ADT plug-in. It defines your application's build target and other build system options, as required. 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/androidbook/myfirstandroidapp/MyFirstAndroidAppActivity.java`**—Main entry point to this application, named `MyFirstAndroidAppActivity`. This activity has been defined as the default launch activity in the Android manifest file.
- **`/gen/com/androidbook/myfirstandroidapp/R.java`**—A generated resource management source file. Do not edit this file.
- **`/assets` folder**—Required folder where uncompiled file resources can be included in the project. Application assets are pieces of application data (files, directories) that you do not want managed as application resources.
- **`/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-*`**—Application icon graphic resources are included in several sizes for different device screen resolutions.
- **`/res/layout/main.xml`**—Layout resource file used by `MyFirstAndroidAppActivity` to organize controls on the main application screen.
- **`/res/values/strings.xml`**—The resource file where string resources are defined.

A number of other files are saved on disk as part of the Eclipse project in the workspace. However, the files and resource directories included in the list here are the important project files you will use on a regular basis.

Creating an AVD for Your Project

The next step is to create an AVD that describes what type of device you want to emulate when running the application. For this example, we can use the AVD we created for the Snake application. An AVD describes a device, not an application. Therefore, you can use the same AVD for multiple applications. You can also create similar AVDs with the same configuration but different data (such as different applications installed and different SD card contents).

Creating a Launch Configuration for Your Project

Next, you must create a Run and Debug launch configuration in Eclipse to configure the circumstances under which the MyFirstAndroidApp application builds and launches. The launch configuration is where you configure the emulator options to use and the entry point for your application.

You can create Run configurations and Debug configurations separately, with different options for each. Begin by creating a Run configuration for the application. Follow these steps to create a basic Run configuration for the MyFirstAndroidApp application:

1. Choose Run, Run Configurations (or right-click the project and choose Run As).
2. Double-click Android Application.
3. Name your Run configuration `MyFirstAndroidAppRunConfig`.
4. Choose the project by clicking the Browse button and choosing the MyFirstAndroidApp project.
5. Switch to the Target tab and set the Device Target Selection Mode to Manual.

Tip

If you leave the Device Target Selection Mode set to Automatic when you choose Run or Debug in Eclipse, your application is automatically installed and run on the device if the device is plugged in. Otherwise, the application starts in the emulator with the specified AVD. By choosing Manual, you are always prompted for whether (a) you want your application to be launched in an existing emulator; (b) you want your application to be launched in a new emulator instance and are allowed to specify an AVD; or (c) you want your application to be launched on the device (if it's plugged in). If any emulator is already running, the device is then plugged in, and the mode is set to Automatic, you see this same prompt, too.

Now create a Debug configuration for the application. This process is similar to creating a Run configuration. Follow these steps to create a basic Debug configuration for the MyFirstAndroidApp application:

1. Choose Run, Debug Configurations (or right-click the project and choose Debug As).
2. Double-click Android Application.
3. Name your Debug configuration `MyFirstAndroidAppDebugConfig`.

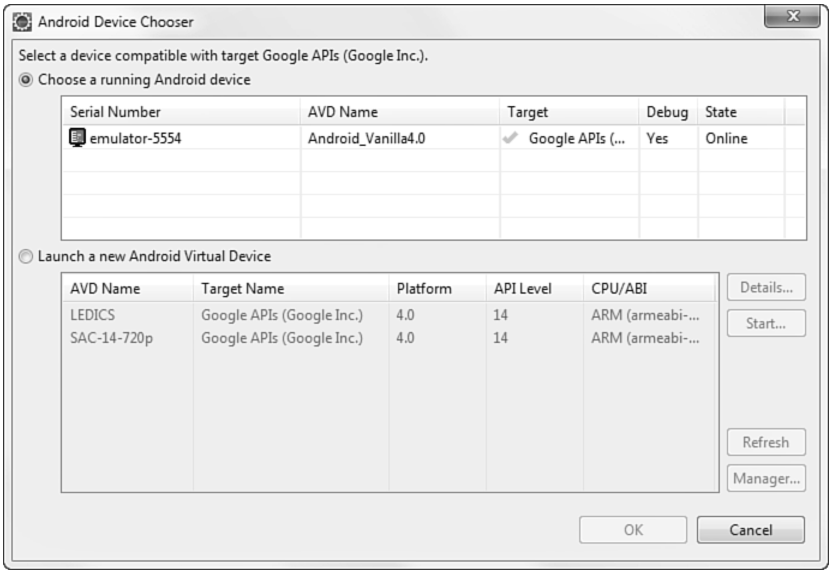
4. Choose the project by clicking the Browse button and choosing the MyFirstAndroidApp project.
5. Switch to the Target tab and set the Device Target Selection Mode to Manual.
6. Click Apply and then click Close.

You now have a Debug configuration for your application.

Running Your Android Application in the Emulator

Now you can run the MyFirstAndroidApp application using the following steps:

1. Choose the Run As icon drop-down menu on the toolbar (▶ ▾).
2. Pull the drop-down menu and choose the Run configuration you created. (If you do not see it listed, choose the Run Configurations... item and select the appropriate configuration. The Run configuration shows up on this drop-down list the next time you run the configuration.)
3. Because you chose the Manual Target Selection mode, you are now prompted for your emulator instance. Change the selection to Launch a New Android Virtual Device and then select the AVD you created, as shown in Figure 3.13. Here you can choose from an already-running emulator or launch a new instance with an AVD that is compatible with the application settings.


Android Device Chooser

Select a device compatible with target Google APIs (Google Inc.).

● Choose a running Android device

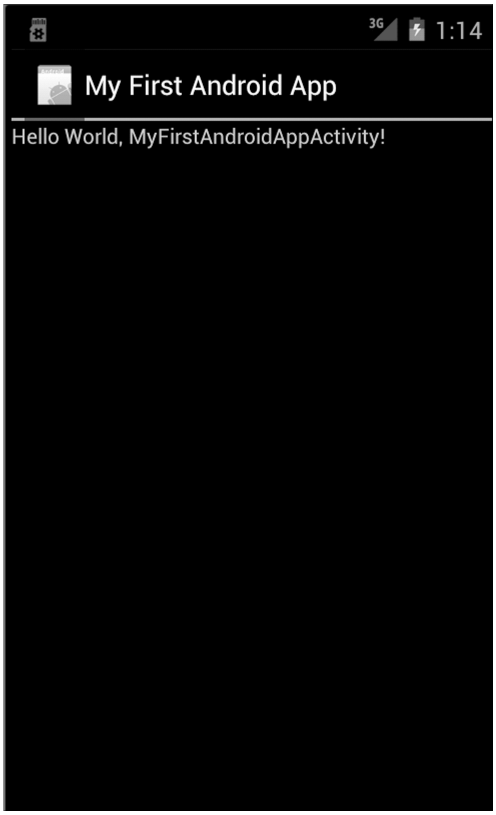
Serial Number	AVD Name	Target	Debug	State
emulator-5554	Android_Vanilla4.0	✓ Google APIs (...	Yes	Online

○ Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level	CPU/ABI
LEDICS	Google APIs (Google Inc.)	4.0	14	ARM (armeabi-...
SAC-14-720p	Google APIs (Google Inc.)	4.0	14	ARM (armeabi-...

Figure 3.13 Manually choosing a target selection mode.

4. The Android emulator starts up, which might take a moment.
5. Click the Menu button or push the slider to the right to unlock the emulator.
6. The application starts, as shown in Figure 3.14.



Hello World, MyFirstAndroidAppActivity!

Figure 3.14 My First Android App running in the emulator.

7. Click the Back button in the Emulator to end the game or click Home to suspend it.
8. Click the grid button to browse all installed applications. Your screen looks something like Figure 3.15.
9. Click the My First Android Application icon to launch the application again.

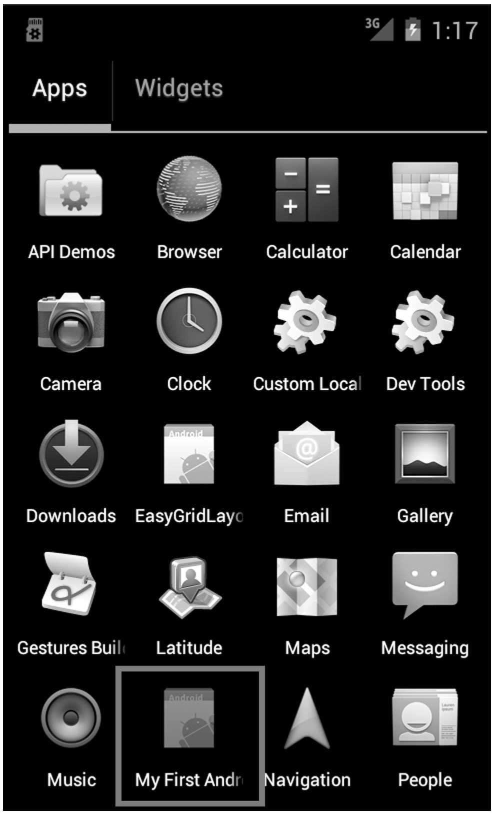


Figure 3.15 The My First Android App icon shown in the application listing.

Debugging Your Android Application in the Emulator

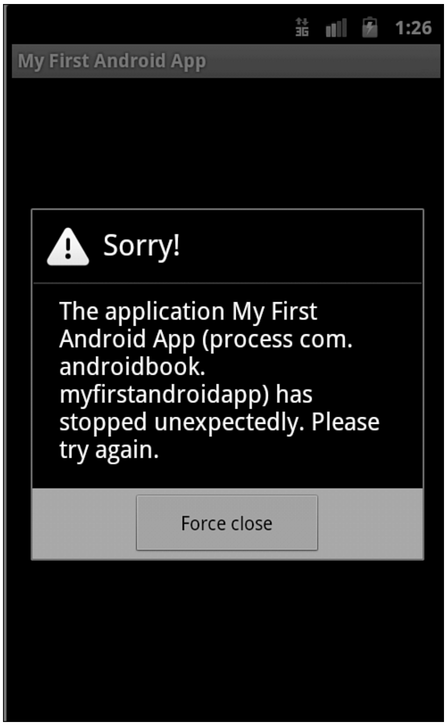
Before we go any further, you need to become familiar with debugging in the emulator. To illustrate some useful debugging tools, let's manufacture an error in the My First Android Application.

In your project, edit the source file called `MyFirstAndroidApp.java`. Create a new method called `forceError()` in your class and make a call to this method in your `Activity` class's `onCreate()` method. The `forceError()` method forces a new unhandled error in your application.

The `forceError()` method should look something like this:

```
public void forceError() {
    if(true) {
        throw new Error("Whoops");
    }
}
```

It's probably helpful at this point to run the application and watch what happens. Do this using the Run configuration first. In the emulator, you see that the application has stopped unexpectedly. You are prompted by a dialog that enables you to forcefully close the application, as shown in Figure 3.16.



The application My First Android App (process com.androidbook.myfirstandroidapp) has stopped unexpectedly. Please try again.

Figure 3.16 My First Android App crashing gracefully.

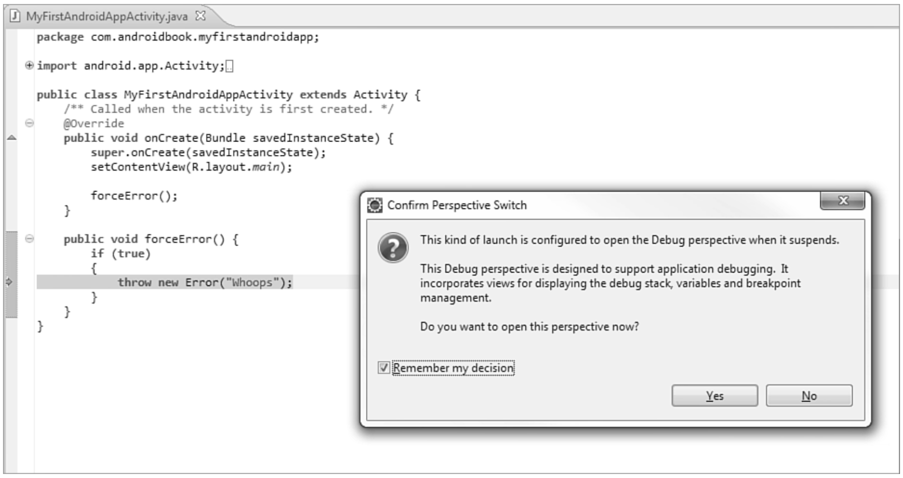
Shut down the application but keep the emulator running. Now it's time to debug. You can debug the MyFirstAndroidApp application using the following steps:

1. Choose the Debug As icon drop-down menu on the toolbar.
2. Pull the drop-down menu and choose the Debug configuration you created. (If you do not see it listed, choose the Debug Configurations... item and select the

appropriate configuration. The Debug configuration shows up on this drop-down list the next time you run the configuration.)

3. Continue as you did with the Run configuration and choose the appropriate AVD and then launch the emulator again, unlocking it if needed.

It takes a moment for the debugger to attach. If this is the first time you've debugged an Android application, you may need to click through some dialogs, such as the one shown in Figure 3.17, the first time your application attaches to the debugger.



```
package com.androidbook.myfirstandroidapp;

import android.app.Activity;

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

        forceError();
    }

    public void forceError() {
        if (true)
        {
            throw new Error("Whoops");
        }
    }
}
```

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

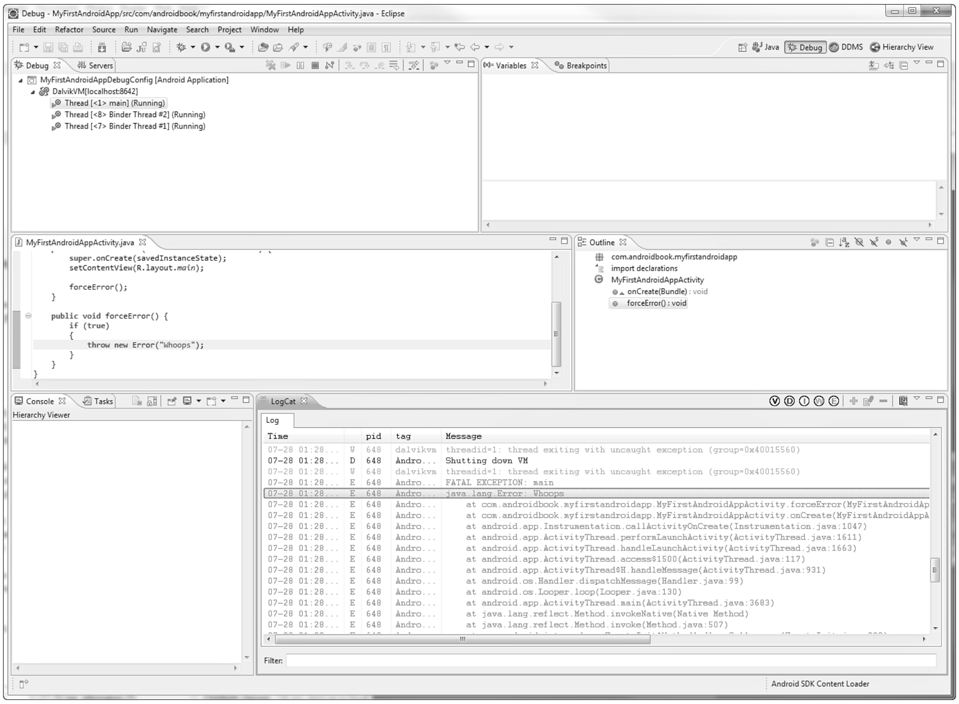
Do you want to open this perspective now?

[✓] Remember my decision

Figure 3.17 Switching debug perspectives for Android emulator debugging.

In Eclipse, use the Debug perspective to set breakpoints, step through code, and watch the LogCat logging information about your application. This time, when the application fails, you can determine the cause using the debugger. You might need to click through several dialogs as you set up to debug within Eclipse. If you allow the application to continue after throwing the exception, you can examine the results in the Debug perspective of Eclipse. If you examine the LogCat logging pane, you see that your application was forced to exit due to an unhandled exception (see Figure 3.18).

Specifically, there's a red `AndroidRuntime` error: `java.lang.Error: Whoops`. Back in the emulator, click the Force Close button. Now set a breakpoint on the `forceError()` method by right-clicking the left side of the line of code and choosing Toggle Breakpoint (or pressing Ctrl+Shift+B).


Debug - MyFirstAndroidApp/src/com/androidbook/myfirstandroidapp/MyFirstAndroidAppActivity.java - Eclipse
File Edit Refactor Source Run Navigate Search Project Window Help
Java
Debug
DDMS
Hierarchy View
Debug
Servers
MyFirstAndroidAppDebugConfig [Android Application]
DalvikVM[localhost:8642]
Thread [<1> main] (Running)
Thread [<8> Binder Thread #2] (Running)
Thread [<7> Binder Thread #1] (Running)
Variables
Breakpoints
MyFirstAndroidAppActivity.java
Outline
com.androidbook.myfirstandroidapp
import declarations
MyFirstAndroidAppActivity
onCreate(Bundle) : void
forceError() : void
Console
Tasks
Hierarchy Viewer
LogCat
Log
Filter:
Android SDK Content Loader

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        forceError();
    }

    public void forceError() {
        if (true)
        {
            throw new Error("Whoops");
        }
    }
}
```

```
Time                pid  tag       Message
07-28 01:28...  W   648  dalvikvm  threadid=1: thread exiting with uncaught exception (group=0x40015560)
07-28 01:28...  D   648  Andro...  Shutting down VM
07-28 01:28...  W   648  dalvikvm  threadid=1: thread exiting with uncaught exception (group=0x40015560)
07-28 01:28...  E   648  Andro...  FATAL EXCEPTION: main
07-28 01:28...  E   648  Andro...  java.lang.Error: Whoops
07-28 01:28...  E   648  Andro...     at com.androidbook.myfirstandroidapp.MyFirstAndroidAppActivity.forceError(MyFirstAndroidAp
07-28 01:28...  E   648  Andro...     at com.androidbook.myfirstandroidapp.MyFirstAndroidAppActivity.onCreate(MyFirstAndroidAppA
07-28 01:28...  E   648  Andro...     at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1047)
07-28 01:28...  E   648  Andro...     at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1611)
07-28 01:28...  E   648  Andro...     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:1663)
07-28 01:28...  E   648  Andro...     at android.app.ActivityThread.access$1500(ActivityThread.java:117)
07-28 01:28...  E   648  Andro...     at android.app.ActivityThread$H.handleMessage(ActivityThread.java:931)
07-28 01:28...  E   648  Andro...     at android.os.Handler.dispatchMessage(Handler.java:99)
07-28 01:28...  E   648  Andro...     at android.os.Looper.loop(Looper.java:130)
07-28 01:28...  E   648  Andro...     at android.app.ActivityThread.main(ActivityThread.java:3683)
07-28 01:28...  E   648  Andro...     at java.lang.reflect.Method.invokeNative(Native Method)
07-28 01:28...  E   648  Andro...     at java.lang.reflect.Method.invoke(Method.java:507)
```

Figure 3.18 Debugging My First Android App in Eclipse.

Tip

In Eclipse, you can step through code using Step Into (F5), Step Over (F6), Step Return (F7), or Resume (F8). On Mac OS X, you might find that the F8 key is mapped globally. If you want to use the keyboard convenience command, you might want to change the keyboard mapping in Eclipse by choosing Eclipse, Preferences, General, Keys and then finding the entry for Resume and changing it to something else. Alternatively, you can change the Mac OS X global mapping by going to System Preferences, Keyboard & Mouse, Keyboard Shortcuts and then changing the mapping for F8 to something else.

In the emulator, restart your application and step through your code. You see that your application has thrown the exception and then the exception shows up in the Variable Browser pane of the Debug Perspective. Expanding its contents shows that it is the "Whoops" error.

This is a great time to crash your application repeatedly and get used to the controls. While you're at it, switch over to the DDMS perspective. You note the emulator has a list of processes running on the device, such as `system_process` and `com.android.phone`. If you launch `MyFirstAndroidApp`, you see `com.androidbook.myfirstandroidapp` show up as a process on the emulator listing. Force the app to close because it crashes, and note that it disappears from the process list. You can use DDMS to kill processes, inspect threads and the heap, and access the phone file system.

Adding Logging Support to Your Android Application

Before you start diving into the various features of the Android SDK, you should familiarize yourself with logging, a valuable resource for debugging and learning Android. Android logging features are in the `Log` class of the `android.util` package. Some helpful methods in the `android.util.Log` class are shown in Table 3.1.

Table 3.1 **Commonly Used Logging Methods**

Method	Purpose
`Log.e()`	Log errors
`Log.w()`	Log warnings
`Log.i()`	Log informational messages
`Log.d()`	Log debug messages
`Log.v()`	Log verbose messages

To add logging support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statement for the `Log` class:

```
import android.util.Log;
```

Tip

To save time in Eclipse, you can use the imported classes in your code and add the imports needed by hovering over the imported class name and choosing the Add Imported Class option.

You can also use the Organize Imports command (Ctrl+Shift+O in Windows or Command+Shift+O on a Mac) to have Eclipse automatically organize your imports. This removes unused imports and adds new ones for packages used but not imported. If a naming conflict arises, as it often does with the `Log` class, you can choose the package you intended to use.

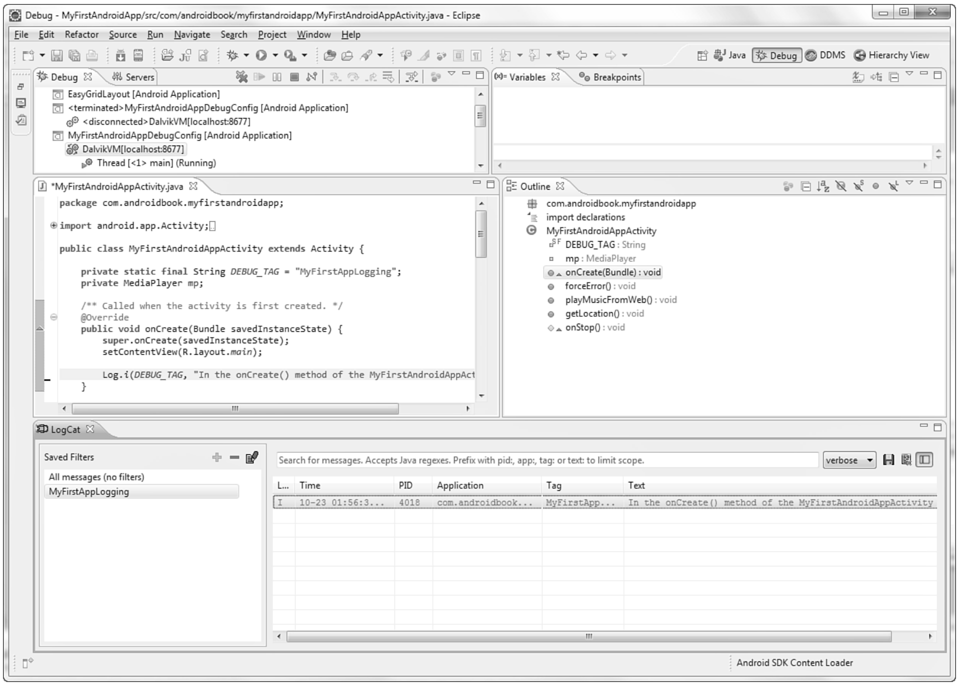
Next, within the `MyFirstAndroidApp` class, declare a constant string that you use to tag all logging messages from this class. You can use the LogCat utility within Eclipse to filter your logging messages to this `DEBUG_TAG` tag string:

```
private static final String DEBUG_TAG= "MyFirstAppLogging";
```

Now, within the `onCreate()` method, you can log something informational:

```
Log.i(DEBUG_TAG,
    "In the onCreate() method of the MyFirstAndroidAppActivity Class");
```

While you're here, you must comment out your previous `forceError()` call so that your application doesn't fail. Now you're ready to run `MyFirstAndroidApp`. Save your work and debug it in the emulator. You notice that your logging messages appear in the LogCat listing, with the Tag field MyFirstAppLogging (see Figure 3.19).


Debug - MyFirstAndroidApp/src/com/androidbook/myfirstandroidapp/MyFirstAndroidAppActivity.java - Eclipse
File Edit Refactor Source Run Navigate Search Project Window Help
Java Debug DDMS Hierarchy View
Debug Servers
EasyGridLayout [Android Application]
<terminated>MyFirstAndroidAppDebugConfig [Android Application]
<disconnected>DalvikVM[localhost:8677]
MyFirstAndroidAppDebugConfig [Android Application]
DalvikVM[localhost:8677]
Thread [<1> main] (Running)
Variables Breakpoints
*MyFirstAndroidAppActivity.java
Outline
com.androidbook.myfirstandroidapp
import declarations
MyFirstAndroidAppActivity
DEBUG_TAG : String
mp : MediaPlayer
onCreate(Bundle) : void
forceError() : void
playMusicFromWeb() : void
getLocation() : void
onStop() : void
LogCat
Saved Filters
All messages (no filters)
MyFirstAppLogging
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose
Android SDK Content Loader

```
package com.androidbook.myfirstandroidapp;

import android.app.Activity;

public class MyFirstAndroidAppActivity extends Activity {

    private static final String DEBUG_TAG = "MyFirstAppLogging";
    private MediaPlayer mp;

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

        Log.i(DEBUG_TAG, "In the onCreate() method of the MyFirstAndroidAppAct
    }
```

```
L...  Time            PID   Application         Tag             Text
I     10-23 01:56:3...  4018  com.androidbook...  MyFirstApp...   In the onCreate() method of the MyFirstAndroidAppActivity
```

Figure 3.19 A LogCat log for My First Android App.

Adding Some Media Support to Your Application

Next, let's add some pizzazz to `MyFirstAndroidApp` by having the application play an MP3 music file. Android media player features are found in the `MediaPlayer` class of the `android.media` package.

You can create `MediaPlayer` objects from existing application resources or by specifying a target file using a Uniform Resource Identifier (URI). For simplicity, we begin by accessing an MP3 using the `Uri` class from the `android.net` package.

Some methods in the `android.media.MediaPlayer` and `android.net.Uri` classes are shown in Table 3.2.

Table 3.2 **Commonly Used MediaPlayer and URI Parsing Methods**

Method	Purpose
`MediaPlayer.create()`	Creates a new media player with a given target to play
`MediaPlayer.start()`	Starts media playback
`MediaPlayer.stop()`	Stops media playback
`MediaPlayer.release()`	Releases the media player resources
`Uri.parse()`	Instantiates a `Uri` object from an appropriately formatted URI address

To add MP3 playback support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statements for the `MediaPlayer` class:

```
import android.media.MediaPlayer;
import android.net.Uri;
```

Next, within the `MyFirstAndroidApp` class, declare a member variable for your `MediaPlayer` object:

```
private MediaPlayer mp;
```

Now, create a new method called `playMusicFromWeb()` in your class and make a call to this method in your `onCreate()` method. The `playMusicFromWeb()` method creates a valid `Uri` object, creates a `MediaPlayer` object, and starts the MP3 playing. If the operation should fail for some reason, the method logs a custom error with your logging tag. The `playMusicFromWeb()` method should look something like this:

```
public void playMusicFromWeb() {
    try {
        Uri file = Uri.parse("http://www.perlgurl.org/podcast/archives"
            + "/podcasts/PerlgurlPromo.mp3");
        mp = MediaPlayer.create(this, file);
        mp.start();
    }
    catch (Exception e) {
        Log.e(DEBUG_TAG, "Player failed", e);
    }
}
```

As of Android 4.0 (API Level 14), using the `MediaPlayer` class to access media content on the Web requires the `INTERNET` permission to be registered in the application's Android manifest file. Finally, your application requires special permissions to access location-based functionality. You must register this permission in your `AndroidManifest.xml` file. To add permissions to your application, perform the following steps:

1. Double-click the AndroidManifest.xml file.
2. Switch to the Permissions tab.
3. Click the Add button and choose Uses Permission.
4. In the right pane, select `android.permission.INTERNET`.
5. Save the file.

Later on, you'll learn all about the various `Activity` states and callbacks that could contain portions of the `playMusicFromWeb()` method. For now, know that the `onCreate()` method is called every time the user navigates to the `Activity` (forward or backward) and whenever he or she rotates the screen or causes other device configuration changes. This doesn't cover all cases, but will work well enough for this example.

And finally, you want to cleanly exit when the application shuts down. To do this, you need to override the `onStop()` method of your `Activity` class and stop the `MediaPlayer` object and release its resources. The `onStop()` method should look something like this:

```
protected void onStop() {
    if (mp != null) {
        mp.stop();
        mp.release();
    }
    super.onStop();

}
```

Tip

In Eclipse, you can right-click within the class and choose Source (or press Alt+Shift+S). Choose the option Override/Implement Methods and select the `onStop()` method.

Now, if you run `MyFirstAndroidApp` in the emulator (and you have an Internet connection to grab the data found at the URI location), your application plays the MP3. When you shut down the application, the `MediaPlayer` is stopped and released appropriately.

Adding Location-Based Services to Your Application

Your application knows how to say "Hello" and play some music, but it doesn't know where it's located. Now is a good time to become familiar with some simple location-based calls to get the GPS coordinates. To have some fun with location-based services and maps integration, you will use some of the Google applications available on typical Android devices—specifically, the Maps application. You do not need to create another AVD, because you included the Google APIs as part of the target for the AVD you already created.

Configuring the Location of the Emulator

The emulator does not have location sensors, so the first thing you need to do is seed your emulator with some GPS coordinates. You can find the exact steps for how to do this in Appendix A, in the section "Configuring the GPS Location of the Emulator." After you have configured the location of your emulator, the Maps application should now display your simulated location, as shown in Figure 3.20.

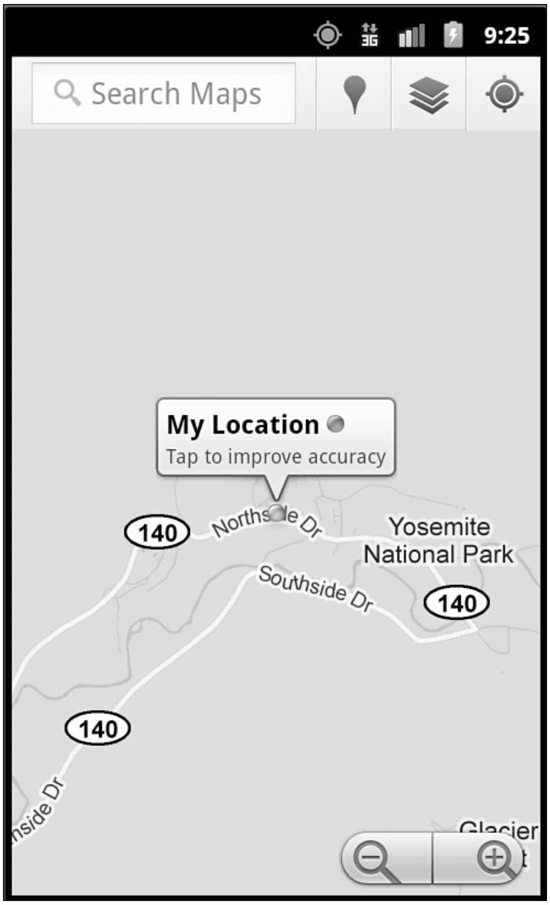


Figure 3.20 Setting the location of the emulator to Yosemite Valley.

Your emulator now has a simulated location: Yosemite Valley!

Finding the Last Known Location

To add location support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statements:

```
import android.location.Location;
import android.location.LocationManager;
```

Now, create a new method called `getLocation()` in your class and make a call to this method in your `onCreate()` method. The `getLocation()` method gets the last known location on the device and logs it as an informational message. If the operation fails for some reason, the method logs an error.

The `getLocation()` method should look something like this:

```
public void getLocation() {
    try {
        LocationManager locMgr = (LocationManager)
            getSystemService(LOCATION_SERVICE);
        Location recentLoc = locMgr.
            getLastKnownLocation(LocationManager.GPS_PROVIDER);
        Log.i(DEBUG_TAG, "loc: " + recentLoc.toString());
    }
    catch (Exception e) {
        Log.e(DEBUG_TAG, "Location failed", e);
    }
}
```

Finally, your application requires special permissions to access location-based functionality. You must register this permission in your `AndroidManifest.xml` file. To add location-based service permissions to your application, perform the following steps:

1. Double-click the `AndroidManifest.xml` file.
2. Switch to the Permissions tab.
3. Click the Add button and choose Uses Permission.
4. In the right pane, select `android.permission.ACCESS_FINE_LOCATION`.
5. Save the file.

Now, if you run My First Android App in the emulator, your application logs the GPS coordinates you provided to the emulator as an informational message, viewable in the LogCat pane of Eclipse.

Debugging Your Application on the Hardware

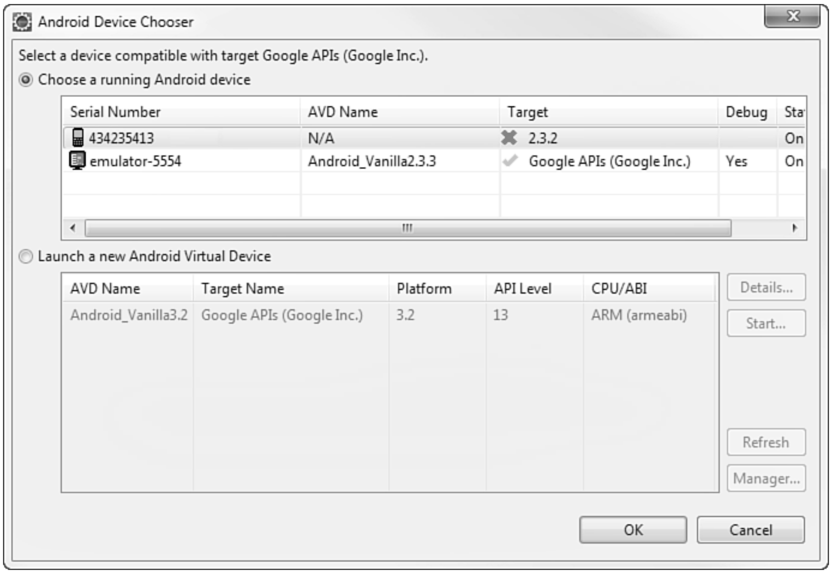
You mastered running applications in the emulator. Now let's put the application on real hardware. First, you must register your application as debuggable in your `AndroidManifest.xml` file. To do this, perform the following steps:

1. Double-click the `AndroidManifest.xml` file.
2. Change to the Application tab.
3. Set the Debuggable application attribute to true.
4. Save the file.

You can also modify the `application` element of the `AndroidManifest.xml` file directly with the `android:debuggable` attribute, as shown here:

```
<application ... android:debuggable="true">
```

Now, connect an Android device to your computer via USB and re-launch the Run configuration or Debug configuration of the application. Because you chose Manual mode for the configuration, you should now see a real Android device listed as an option in the Android Device Chooser (see Figure 3.21).



Select a device compatible with target Google APIs (Google Inc.).

[✓ Choose a running Android device]

Serial Number	AVD Name	Target	Debug	Sta
434235413	N/A	✗ 2.3.2		On
emulator-5554	Android_Vanilla2.3.3	✓ Google APIs (Google Inc.)	Yes	On

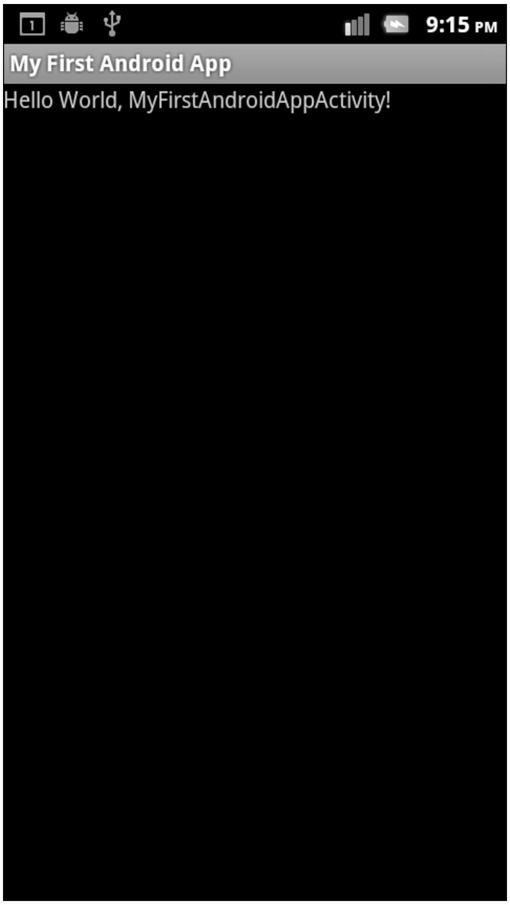
Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level	CPU/ABI
Android_Vanilla3.2	Google APIs (Google Inc.)	3.2	13	ARM (armeabi)

Figure 3.21 Android Device Chooser with USB-connected Android device.

Choose the Android device as your target, and you see that the My First Android App application gets loaded onto the Android device and launched, just as before. Provided you have enabled the development debugging options on the device, you can debug the application here as well. You can tell the device is actively using a USB debugging connection, because a little Android bug-like icon appears in the notification bar (). Figure 3.22 shows a screenshot of the application running on a real device (in this case, a smartphone running Android 2.3.2).

Debugging on the device is much the same as debugging on the emulator, but with a couple of exceptions. You cannot use the emulator controls to do things such as send an SMS or configure the location to the device, but you can perform real actions (true SMS, actual location data) instead.



Hello World, MyFirstAndroidAppActivity!

Figure 3.22 My First Android App running on Android device hardware.

Summary

This chapter showed you how to add, build, run, and debug Android projects using Eclipse. You started by testing your development environment using a sample application from the Android SDK and then you created a new Android application from scratch using Eclipse. You also learned how to make some quick modifications to the application, demonstrating some exciting Android features you learn about in future chapters.

In the next few chapters, you learn about the tools available for use developing Android applications and then focus on the finer points about defining your Android application using the application manifest file and how the application lifecycle works. You also learn how to organize your application resources, such as images and strings, for use within your application.

References and More Information

Android SDK Reference regarding the application `Activity` class:
http://d.android.com/reference/android/app/Activity.html
Android SDK Reference regarding the application `Log` class:
http://d.android.com/reference/android/util/Log.html
Android SDK Reference regarding the application `MediaPlayer` class:
http://d.android.com/reference/android/media/MediaPlayer.html
Android SDK Reference regarding the application `Uri` class:
http://d.android.com/reference/android/net/Uri.html
Android SDK Reference regarding the application `LocationManager` class:
http://d.android.com/reference/android/location/LocationManager.html
Android Dev Guide: "Developing on a Device":
http://d.android.com/guide/developing/device.html
Android Resources: "Common Tasks and How to Do Them in Android":
http://d.android.com/resources/faq/commontasks.html
Android Sample Code: "Snake":
http://d.android.com/resources/samples/Snake

This page intentionally left blank

Index

Symbols

# (hash symbols), 151

(…) (ellipsis), 174

A

above attribute, 213

AbsoluteLayout class, 214

abstracted LCD density, 407

accelerometer support, 406

accessing

Booleans, 150
browser information, 289–290
calendar data, 291
call logs, 288–289
colors, 151
contacts, 292
content providers, 286
device settings, 292
dictionary, 291
dimensions, 153
directories, 105
drawables, 154
external storage, 284
files, 105, 278
images, 155
integers, 150
layouts, 166
media, 286–288

menus, 159
nine-patch stretchable graphics, 155
preferences, 105, 267–268
raw files, 161
resources, 142
screen information, 304
strings, 147–148
tweened animations, 158
voicemail, 291
XML files, 160

activities
application navigation, 115–116
avoiding killing, 109–110
callbacks, 107–109
initializing/retrieving activities, 109
static data, initializing, 108
stopping/saving/releasing, 109
defined, 103
dialogs, adding, 254
fragments
Activity classes, creating, 246–247
applications, designing, 238
attaching, 237
compatibility, 305
creating, 237
defined, 104, 111
defining, 235
destroying, 237
detaching, 237
dialogs, 238, 257–260
layout resource files, creating, 244–246
legacy support, 247–248
lifecycle, 235
ListFragment, implementing, 240–243
lists, 238
MP3 music player example, 111–113
overview, 233
pausing, 237
resuming activity, 237
screen workflow, 234
specialty, 237
stopping activity, 237
updates, 236
user preferences, 238
visibility, 237
web content, 238
website, 118
WebView, implementing, 243–244
workflow flexibility, improving, 111–113
game application example, 106
intents
action/data, 114
additional information, adding, 115
application navigation, 115–116
broadcasting, 117
filters, 114, 132–133
Google common, 115
launching activities by class name, 113
launching external activities, 114
primary entry points, choosing, 132
website, 118
launch, creating, 64
lifecycle, 106–107
registering, 131–133
src/com/androidbook/myfirstandroidapp/MyFirstAndroidAppActivity.java, 65
stacks, 107
state, saving into bundles, 110
static data, destroying, 110

Activity class
- callbacks, 107-109
  - onCreate( ), 108
  - onPause( ), 109
  - onResume( ), 109
- defined, 103
- website, 81, 118

ActivityUnitTestCase class, 359

ad-hoc permissions, 31

ad revenue, 379

adapters
- AdapterView class, 222-224
- arrays, 222-223
- classes, 222
- creating, 222
- databases, 223-224
- defined, 222
- lists, assigning, 225

AdapterView class, 222-224

ADB (Android Debug Bridge), 87, 100

addFooterView( ) method, 226

addHeaderView( ) method, 226

adding
- additional activity information with intents, 115
- applications on Home screen, 419
- contacts, 297-298
- dialogs
  - activities, 254
  - fragment-based method, 252
  - legacy method, 251
- location-based services, 76-78
- logging support, 73-74
- media support, 74, 76
- MP3 playback, 75
- pages, 419
- permissions, 75
- preferences, 264-267
- progress bars to title bars, 193
- records, 297-298
- sample projects, 52
- statistics collection, 379
- Support (Compatibility) Package, 248-249, 305
- View controls to ViewGroup, 204

addPreferencesFromResource( ) method, 271

addTab( ) method, 227, 229

addToBackStack( ) method, 236

addView( ) method, 204

addWord( ) method, 291

Adobe AIR website, 33

ADT Eclipse plug-in, 46-47, 88
- resources
  - creating, 143-146
  - editors, 89
  - layout resource editor, 164-166
- UI designer, 90

advantages
- Android, 22, 25
- application development, 26
- application integration, 26
- development tools, 25
- free market, 27-28
- IDEs, 25
- Java, 26
- open-source, 25
- publication, 27
- SDKs, 25

AlarmClock content provider, 285, 300

alert dialogs, 252

AlertDialog class, 252, 260

aliases (resources), 162

alignBottom attribute, 213

alignLeft attribute, 212

alignParentBottom attribute, 212
alignParentLeft attribute, 212
alignParentRight attribute, 212
alignParentTop attribute, 212
alignRight attribute, 212
alignTop attribute, 213
alternative marketplaces, 393
alternative resources, 141-142
  benefits, 308
  creating, 309
  data, saving, 318
  defined, 308
  directory qualifiers, 309
    bad examples, 311
    case, 310
    combining, 310
    customizing, 311
    default resources, including, 311
    good examples, 311
    limits, 310
    listing of, 311–313
    requirements, 311
  icons example, 310
  organization schemes, 317
  performance issues, 317
  requesting, 316
  runtime changes, handling, 318
  screen orientations, 316
  website, 321
Amazon Appstore, 393
analog clocks, 197
AnalogClock control, 197
anddev.org forum, 7
Android
  advantages, 22
  command-line tool, 98
  core files, 65
  Debug Bridge (ADB), 87, 100
  Developer resources
    blog, 350
    website, 6, 35
  documentation, 83–85
    maintenance/support, 345
    mobile development projects, 337–338
    SDK, 43
    sections, 83
    website, 83
  licensing, 25
  manifest file, 119
  mascot, 24
  overview, 5
  project, 20
  Project Wizard, 62–64
    application information, 64
    build targets, 63
    file locations, choosing, 63
    launch activities, 64
    minimum SDK version, 64
    names, 62
    package names, 64
  Support (Compatibility) Package, 247
  Tools Project Site website, 7
  Virtual Devices (AVDs), 56
    creating, 65, 403–404
    display options, 404–405
    emulator, 401, 410
    hardware configuration settings, 405–407
    management website, 421
    Manager, 47–48, 402
    Snake application, creating, 56–57
  Virtual Device Manager (AVD Manager), 47–48, 402

Android Market, 385, 392
alternatives, 393-394
applications
removing, 393
upgrading, 392
uploading, 387
contact/consent options, 390
developer accounts, creating, 385-387
fees, 390
filters, 381, 395
help, 390
listing details, 388
marketing assets, uploading, 388
packaging requirements, 381
publishing options, 390
return policies, 392
reviews, 344
website, 7, 385, 395
android package, 44
AndroidManifest.xml file, 65
<animation-list> tag, 157
AnimationDrawable resources, 157
animations
frame-by-frame, 156-157
storing, 139
tweened, 156
accessing, 158
defining, 157
overview, 157
spinning graphic example, 158
types, 156
anonymous diagnostics, gathering, 354
APIs (application programming interfaces), 32
ApiDemos sample app, 51
C2DM, 45
Google, 45-46
legacy support, 45
levels website, 136
SDK versions, 127
third-party, 45
XML SAX support, 45
App Widgets, 172
application framework
important packages, 43
third-party APIs, 45
application programming interfaces. *See* APIs
Application tab (Eclipse), 121
applications
Android Market, managing
removing, 393
return policies, 392
reviews, 344
upgrades, 392
assets, retrieving, 105
availability, 22
creating
AVD, 65
core files, 65
device target selection mode, 66
launch configuration, 66-67
new project, creating, 62-64
data
directories, 278-280
storing, 275-276
debugging
ADB, 87
emulator, 69-73
DDMS, 87
devices, 78-79

designing
  best practices, 355, 361
  billing/revenue generation, 351-352
  code diagnostics, 353, 358-359
  code reviews, 358
  coding standards, 357
  feasibility, testing, 356
  mistakes, avoiding, 355, 360
  rules, 347-348
  security, 351
  single device bugs, 359
  software process, choosing, 356
  stability/responsiveness, 349-350
  third-party standards, 352
  tools, 354, 360
  updates/upgrades, 354
  user demands, meeting, 348
  user interfaces, 348-349
development advantages, 26
feasibility assessments, 336
Flash support, 33
fragment-based, designing, 238
Google involvement, 22
Home screen, adding, 419
icons
  configuring, 125
  setting, 380
in-application billing, 379
installations, testing, 372
integration, 26
interoperability, 342
legacy support, 247-248
lifecycle, 34
location-based services, 77-78
manifest file settings, 121
names
  configuring, 125
  setting, 380
native versus third-party, 33
navigation, 115-116
network-driven, 340
operating system/hardware interaction, 34
PeakBagger 1.0, 86
performance, testing, 373
popular, creating, 374
preferences
  accessing, 105, 267-268
  adding, 264
  appropriateness, 263
  data types supported, 264
  editing, 266-267
  overview, 263
  private, 264
  reacting, 267
  reading, 265
  searching, 265
  shared, 265
release versions, testing, 384
resources, retrieving, 105
rs:ResEnum, 168
running
  MyFirstAndroidApp, 67-68
  as operating system users, 31
sample, 50-51
signing, 382-384
Snake
  adding to Eclipse, 54
  AVD, creating, 56-57

launch configuration, creating, 58-59
missing folder error, 56
running in emulator, 59-60
sample code website, 81
standalone, 340
statistics collection, 379
traceview, 375
upgrades, testing, 371
uploading to Android Market, 387
usability, testing, 370
versioning, 125
visual appeal, 370
web, 33

ApplicationTestCase class, 359

apply( ) method, 267

architecture, 29
Dalvik VM, 31
Linux operating system, 30

ArrayAdapter class, 222-223

arrays
adapters, 222-223
integers, 151
accessing, 150
creating, 151
defining, 150
storing, 139
mixed-type, storing, 139
strings, 149
accessing, 147-148
bold/italic/underline, 147
formatting, 146-147
overview, 146
plural, 149
storing, 139
<string> tag, 146

aspect ratio (screens), 313

assets
folder, 65
retrieving, 105

attributes
AutoCompleteTextView control, 181
DatePicker control, 190
debuggable, 382
EditText control, 177
FrameLayout class, 207-209
GridLayout class, 216
layout, 205-206
LinearLayout class, 210
Preference class, 269
RelativeLayout class, 211
TableLayout class, 214
TextView control, 173-174
autoLink, 174-176
ellipsize, 174
ems, 174
lines, 174

audio playback/recording support, 406

AutoCompleteTextView class, 179-181

autocompletion, 179-181. 443

autoLink attribute, 174-176

automating testing, 368

AVDs (Android Virtual Devices), 56
creating, 65, 403-404
display options, 404-405
emulator
launching, 411
settings, 401
hardware configuration settings, 405-407
management website, 421
Manager, 47-48, 402
Snake application, creating, 56-57

B

<b> tag, 147
background layout example, 162-163
backup services, testing, 373
battery support, 406
below attribute, 213
best practices
  applications
    popularity, increasing, 374
    upgrades, 371
    usability, 370
  automation, 368
  backup services, 373
  billing, 373
  black-box, 369
  build validations, 368
  conformance, 372
  coverage, maximizing, 367
  defect tracking systems, 363-365
  development
    code diagnostics, 358-359
    code reviews, 358
    coding standards, 357
    feasibility, testing, 356
    mistakes, avoiding, 360
    reference websites, 361
    single device bugs, 359
    software process, choosing, 356
    tools, leveraging, 360
  device upgrades, 372
  emulator, 399-401
    benefits, 368
    disadvantages, 369
  environment, 365-367
  installations, 372
  integration points, 371
  internationalization, 372
  mistakes, 375
  performance, 373
  preproduction devices, 368
  reference websites, 376
  remote servers/services, 370
  third-party standards, 371
  tools, 374-375
  unexpected configuration changes, 373
  visual appeal, 370
  white-box, 369
billing, 351-352
  in-app, 379
  testing, 373
black-box testing, 369
Blog section (documentation), 84
bmgr tool, 98, 375
bold, strings, 147
book website, 6
Booleans, 149
  accessing, 150
  defining, 150
  storing, 139
broadcast receivers, registering, 133
broadcasting intents, 117
Browser content provider, 285, 289-290, 300
browsers
  early WAP, 16
  information, accessing, 289-290
  querying, 290
browsing files, 432
build errors, 447
build targets, choosing, 63
build validations, 368

built-in content providers, 285
bundles activity state, saving, 110
Button class, 183-184
buttons, 183
  basic, 183-184
  check boxes, 183, 186
  image buttons, 185
  radio, 183, 187-189
  switches, 183, 187
  toggle, 183, 186

C

C2DM (Cloud to Device Messaging), 45
cache files, storing, 282-283
cache subdirectory, retrieving, 279
CalendarContract content provider, 285, 291
calendars, accessing, 291
calendarViewShown attribute, 190
callbacks
  activities, 107-109
    initializing/retrieving activities, 109
    static data, initializing, 108
    stopping/saving/releasing, 109
  fragments, 237
calling
  between emulators, 413
  logs, 288-289
CallLog content provider, 285, 288-289, 300
camera support, 406
centerHorizontal attribute, 212
centerInParent attribute, 212
centerVertical attribute, 212
certificate signing, 32
character picker dialogs, 252
CharacterPickerDialog class, 252, 260
check boxes, 183, 186
CheckBox class, 183, 186
CheckBoxPreference class, 274
choosing
  build targets, 63
  devices for tracking, 332
  distribution methods, 377-378
  file locations, 63
  primary entry points, 132
  software processes, 356
    iterative, 327
    waterfall approaches, 326
  source control systems, 339
  versioning schemes, 339-340
Chronometer control, 195-196
circular progress indicators, 192
classes
  AbsoluteLayout, 214
  Activity
    callbacks, 107-109
    defined, 103
    website, 81, 118
  AdapterView, 222-224
  ActivityUnitTestCase, 359
  AlertDialog, 252, 260
  AnalogClock, 197
  AnimationDrawable, 157
  App Widgets, compared, 172
  ApplicationTestCase, 359
  ArrayAdapter, 222-223
  AutoCompleteTextView, 179-181
  Button, 183-184
  CharacterPickerDialog, 252, 260
  CheckBox, 183, 186
  CheckBoxPreference, 274

Chronometer, 195–196
ContactsContract, 296
containers, 221
Context
    defined, 103
    methods, 278–279
    permission modes, 277
    website, 118, 284
creating, 443
CursorAdapter, 222–224
DatePicker, 190–191
DatePickerDialog, 253, 260
Debug, 373
Dialog, 252, 260, 321
DialogFragment, 238, 250, 260
DigitalClock, 196
DisplayMetrics, 304
EditText, 176–179
EditTextPreference, 274
Environment, 284
FieldNoteListFragment, 240–243
FieldNoteViewActivity, 246
FieldNoteWebViewFragment, 243–244
File, 282–284
Fragment
    callback methods, 237
    defined, 104
    reference website, 250
    subclasses, 237
    website, 118
FrameLayout, 207
    attributes, 207–209
    website, 231
Gallery, 231
GalleryView, 222
GridLayout, 216–220, 231
GridView, 222, 231
ImageButton, 185
imports (Eclipse), 73
Intent, 104
LayoutParams, 205
LinearLayout, 209–210, 231
ListActivity, 225, 231
ListFragment, 238
    implementing, 240–243
    reference website, 250
ListPreference, 274
ListView, 222, 231
LocationManager, 81
Log
    methods, 73
    website, 81
MarginLayoutParams, 205
MediaPlayer
    methods, 74
    website, 81
MediaStore, 286
MoreAsserts, 359
MultiAutoCompleteTextView, 181
OnItemClickListener, 224
PerformanceTestCase, 359
Preference, 269, 274
PreferenceActivity, 268–273
PreferenceCategory, 274
PreferenceFragment, 238, 250
PreferenceScreen, 274
ProgressBar, 192–193
ProgressDialog, 253, 260
RadioButton, 183, 187–189

RatingBar, 194-195
RelativeLayout, 211-214, 231
SeekBar, 194
Service, 104, 116
ServiceTestCase, 359
Settings, 292
ShapeDrawable, 153
SharedPreferences, 105, 264
SimpleFragDialogActivity, 258
SlidingDrawer, 230
Spinner, 181-183
StrictMode, 373
Switch, 183, 187
TabActivity, 226-228, 231, 238
TabHost, 226, 229-231
TableLayout, 214-216, 231
TabSpecs, 226
TextView, 173
TimePicker, 191
TimePickerDialog, 253, 260
ToggleButton, 183, 186
TouchUtils, 359
Uri
  methods, 74
  website, 81
View, 171, 204
ViewAsserts, 359
ViewGroup, 204
  layout attributes, 205-206
  layout subclasses, 204
  View containers, 204
  View class, compared, 204
  website, 231
WebView, 243-244
WebViewFragment, 238, 250

clean states (devices), 366-367
clear( ) method, 266
clearCheck( ) method, 189
click events, handling, 224
clients, testing, 337
clocks
  analog, 197
  digital, 196
cloud, testing, 337
Cloud to Device Messaging (C2DM), 45
code
  autocomplete, 443
  build errors, 447
  classes, creating, 443
  comments, 446
  diagnostics, 358-359
  formatting, 444
  imports, 443
  methods, creating, 443
  packaging preparations, 380
    Android Market, 381
    application names/icons, 380
    debugging/logging, turning off, 381
    manifest file, 381
    permissions, 382
    target platforms, 381
    versioning, 380
  QuickFix feature, 446
  refactoring, 444-445
  reorganizing, 446
  reviews, 358
  standards, defining, 357
  stepping through (Eclipse), 72
  versioning, 380
collapseColumns attribute, 214

<color> tag, 151
colors, 151
accessing, 151
defining, 151
formats, 151
storing, 139
column attribute
GridLayout class, 218
TableLayout class, 215
columnCount attribute, 217
columnSpan attribute, 218
command-line tools, 98
bmgr, 98, 375
etc1tool, 99
Exerciser Monkey, 373, 376
layoutopt, 220
logcat, 99, 374, 436-437
mksdcard, 99
sqlite3, 99, 375
zipalign, 99
commands
File menu
New, Android Project, 62
New, Other, 54
Run menu
Debug Configurations, 58, 66
Run Configurations, 66
comments (code), 446
commercializing WAP, 16
commit( ) method, 266
"Common Tasks and How to Do Them in Android" website, 81
commonly used packages, 34
compatibility, 301
alternative resources
benefits, 308
creating, 309
data, saving, 318
defined, 308
directory qualifiers, 309-313
icons example, 310
organization schemes, 317
performance issues, 317
requesting, 316
runtime changes, handling, 318
screen orientations, 316
best practices website, 322
densities, 301
fragments, 305
Google TVs, 319-321
maximizing, 303
nine-patch stretchable graphics, 306
OpenGL versions, 301
platforms, 301
runtime changes, handling, 321
screen
sizes, 301
types, 305-306
Support (Compatibility) Package, 118, 248-249, 305
tablets, 318-319
user interfaces, 303-304
working squares, 306-307
<compatible-screens> tag, 131
competitive hardware, 23
complete platform, 23
completionHint attribute, 181
completionThreshold attribute, 181
Conder, Shane
blog, 35
contacting, 8

configuring
- alternative resource directory qualifiers, 311
- development environment
  - installation, 38
  - software requirements, 37-38
- dialogs, 256
- emulator location, 77
- GL texture compression formats, 131
- hardware for debugging, 39-41
- icons, 125
- launches, 66-67
- operating systems for device debugging, 39
- screen sizes, 130

conformance, testing, 372

consent options (Android Market), 390

console (emulator)
- commands, 419
- connecting, 416
- GPS coordinates, sending, 418
- incoming calls, simulating, 416
- network status, displaying, 418
- power settings, 418
- SMS messages, simulating, 416

contact options (Android Market), 390

contacting Conder/Darcey, 8

contacts
- accessing, 292
- adding, 297-298
- content provider, 292-294
  - permissions, 292
  - querying quickly, 294-295
  - records, 297-299
  - website, 300
- deleting, 298-299
- legacy support, 293-294
- permissions, 292
- querying
  - Contacts provider, 293-294
  - ContactsContract content provider, 295
  - quickly, 294-295
- updating, 298

ContactsContract content provider, 286, 292
- data column classes, 296
- new features, 295
- overview, 295
- permissions, 292
- queries, 295
- querying quickly, 294-295
- website, 297, 300

containers (view), 204, 221
- data-driven, 221-222
  - adapters, 222-225
  - arrays, 222-223
  - click events, handling, 224
  - data, binding, 224
  - databases, 223-224
  - lists of items, 225
- scrolling, 229
- sliding drawers, 230
- switchers, 230
- tabs, 226-228

contains( ) method, 265

content providers, 285
- accessing, 286
- AlarmClock, 285
- Browser, 285, 289-290
- built-in, 285

CalendarContract, 285, 291
CallLog, 285, 288-289
contacts, 292
legacy Contacts provider, 293-294
permissions, 292
querying quickly, 294-295
ContactsContract, 286
data column classes, 296
new features, 295
overview, 295
queries, 295
website, 297
MediaStore, 286-288
audio file titles, retrieving, 287
classes, 286
permissions, 289
records
adding, 297-298
deleting, 298-299
updating, 298
reference websites, 300
registering, 133
SearchRecentSuggestions, 286
Settings, 286, 292
testing, 285
third-party, 299
UserDictionary, 286, 291
VoicemailContract, 286, 291

context
assets, retrieving, 105
defined, 103
files/directories, 105
preferences, 105
resources, 105
retrieving, 104

Context class
defined, 103
methods, 278-279
permission modes, 277
website, 118, 284

contextual links, creating, 174-176

controls, 171. *See also* classes
layout, 172
scrolling, 229
switchers, 230
attributes, 173-174
contextual links, 174-176
height, 174
width, 174

copying files, 432-433

core files, 65

costs, 23

country codes website, 322

coverage, maximizing, 367
application
popularity, increasing, 374
upgrades, 371
usability, 370
automation, 368
backup services, 373
billing, 373
black-box testing, 369
build validations, 368
conformance, 372
device upgrades, 372
emulators
benefits, 368
disadvantages, 369
installations, 372
integration points, 371

internationalization, 372
performance, 373
preproduction devices, 368
remote servers/services, 370
third-party standards, 371
unexpected configuration changes, 373
visual appeal, 370
white-box testing, 369
crashes, monitoring, 345
create( ) method, 75
createTabContent( ) method, 228
creating
AVDs, 65
intents, 114
launch
activities, 64
configurations, 66-67
new projects, 62-64
application information, 64
build targets, 63
file locations, choosing, 63
launch activities, 64
minimum SDK version, 64
names, 62
package names, 64
Nine-Patch Stretchable Graphics, 95-97
resources, 143-146
CursorAdapter class, 222-224
cursors, 223-224
customization development method, 328-329
customizing. *See also* configuring
alternative resource directory qualifiers, 311
development environment
installation, 38
software requirements, 37-38
dialogs, 256
emulator location, 77
GL texture compression formats, 131
icons, 125
launches, 66-67
screen sizes, 130

D

D-Pad support, 406
Dalvik Debug Monitor Server. *See* DDMS
dalvik package, 44
Dalvik VM, 31
dangerous protection level (permissions), 135
Darcey, Lauren
blog, 35
contacting, 8
data
alternative resources, saving, 318
applications
directories, 278
storing, 275-276
devices, storing, 332
intents, 114
preference values, 264
data-driven containers, 221-222
adapters, 222-225
arrays, 222-223
click events, handling, 224
data, binding, 224
databases, 223-224
lists of items, 225
data/app directory, 432

data/data/<package name>/ cache/ directory, 432

data/data/<package name>/ databases/ directory, 432

data/data/<package name>/ directory, 432

data/data/<package name>/ files/ directory, 432

databases, 330

adapters, 223-224

data, storing, 332

devices for tracking, choosing, 332

functions, 332

third-party, 333

DatePicker control, 190-191

DatePickerDialog class, 253, 260

dates

dialogs, 253

picker dialogs, 253

user input, 190-191

DDMS (Dalvik Debug Monitor Server), 47, 87, 423

debuggers, attaching, 425

debugging applications, 87

drag-and-drop operations, 433

Emulator Control pane, 434

features, 424

File Explorer, 430

browsing, 432

copying files, 432-433

deleting files, 433

drag-and-drop operations, 433

garbage collection, 428

heap statistics, monitoring, 427

HPROF files, 429

logging, 436-437

memory allocations, 430

perspective (Eclipse), 47

processes, stopping, 426

screen captures, 435

standalone application, 423

threads, monitoring, 426

website, 100

Debug class, 373

Debug Configurations command (Run menu), 58, 66

debuggable attribute, 382

debugging. *See also* troubleshooting

ADB, 87

configurations, 66, 409

devices, 78-79

Eclipse

perspective, 47

stepping through code, 72

emulator, 69-73

hardware, enabling, 39-41

MyFirstAndroidApp, 66

operating systems, configuring, 39

turning off, 381

user interfaces

drawing issues, 92-93

layout control organization, 94

websites, 100

default resources, 141-142, 308

defaultValue attribute, 269

defect tracking systems, 363

information, logging, 363

types of defects, 364-365

delete( ) method, 299

deleteFile( ) method, 278

deleting

activity static data, 110

applications from Android Market, 393

contacts, 298-299

dialogs, 255-256
files, 278, 433
preferences, 266-267

densities, compatibility, 301

deprecated methods, 252

designing applications
best practices, 355
code diagnostics, 358-359
code reviews, 358
coding standards, 357
feasibility, testing, 356
mistakes, avoiding, 360
reference websites, 361
single device bugs, 359
software process, choosing, 356
tools, leveraging, 360
billing/revenue generation, 351-352
in-app, 379
testing, 373
diagnostics, leveraging, 353
mistakes, avoiding, 355
rules, 347-348
security, 351
stability/responsiveness, 349-350
third-party standards, 352
tools, 354
updates/upgrades, 354
user demands, meeting, 348
user interfaces, 348-349

destroying. *See* deleting

Dev Guide section (documentation), 84

developers
accounts, creating, 385-387
registration, 32

Developer.com website, 7

"Developing on a Device" website, 81

development environment
configuring
hardware, 39-41
installation, 38
operating systems, 39
software requirements, 37-38
SDK, upgrading, 41
testing, 365
clean states, 366-367
device configurations, 365-366
real world simulations, 367
testing with Snake application, 53
adding to Eclipse, 54
AVD, creating, 56-57
launch configuration, creating, 58-59
running in emulator, 59-60

development tools, 25
ADB, 87
Android command-line tool, 98
Android documentation, 83-85
AVD Manager, 47-48
application design, 354
bmgr, 98, 375
DDMS, 47, 87
debuggers, attaching, 425
debugging applications, 87
drag-and-drop operations, 433
Emulator Control pane, 434
features, 424
File Explorer, 430-433
garbage collection, 428
heap statistics, monitoring, 427
HPROF files, 429
logging, 436-437
memory allocations, 430

perspective (Eclipse), 47
processes, stopping, 426
screen captures, 435
standalone application, 423
threads, monitoring, 426
website, 100
development, leveraging, 360
dmtracedump, 98
Eclipse ADT plug-in, 46-47
creating resources, 143-146
editors, 89
layout resource editor, 164-166
UI designer, 90
emulator, 49
AVDs, 401-407
benefits, 368
best practices, 399-401
calling between, 413
console, 416-419
Control pane, 434
disadvantages, 369
files, managing, 432-433
GPS location, 411-413
Hierarchy Viewer, launching, 92
icon tips, 419
launching, 60, 407-411
limitations, 420-421
location, configuring, 77
messaging between, 415
MyFirstAndroidApp, 67-73
overview, 85, 399
pages, adding, 419
PeakBagger 1.0, 86
performance, 407-408
screen captures, 435
smartphone-style example, 49
Snake app, running, 59-60
startup options, 408
tablet-style example, 50
tips, 419
unlocking, 60
wallpaper, editing, 419
website, 86, 100, 421
etc1tool, 99
Exerciser Monkey, 373, 376
Extract Local Variable, 445
Extract Method, 445
Hierarchy Viewer
launching, 92
layout controls above application content, 92
Layout View mode, 92-93
modes, 92
online tutorial, 94
overview, 91
Pixel Perfect mode, 94
user interface optimization, 94
hprof-conv, 98
layoutopt, 220, 375
logcat, 86-87, 99, 374, 436-437
mksdcard, 99
monkey, 99
monkeyrunner, 99
Nine-Patch Stretchable Graphics, 95-97
Rename, 444
resource editors, 89
sample applications, 50-51
SDK Manager, 47-48
signing, 384

sqlite3, 99
testing, 374–375
traceview, 98
UI designer, 90
website, 98
zipalign, 99

devices
application interaction, 34
availability, 334–335
bugs, handling, 359
clean states, 366–367
compatibility, 301
alternative resources. *See* compatibility, alternative resources
best practices, 322
densities, 301
fragments, 305
Google TVs, 319–321
maximizing, 303
nine-patch stretchable graphics, 306
OpenGL versions, 301
platforms, 301
runtime changes, handling, 321
screen sizes, 301
screen types, 305–306
Support (Compatibility) Package, 305
tablets, 318–319
user interfaces, 303–304
working squares, 306–307
competitive, 23
convergence, 17
databases, 330
data, storing, 332
devices for tracking, choosing, 332
functions, 332
third-party, 333
debugging
configuring, 39–41
enabling, 39–41
MyFirstAndroidApp, 78–79
operating systems, configuring, 39
files
browsing, 432
copying, 432–433
deleting, 433
storing, 332
fragmentation, 365–366
Google Experience, 335
history
Android manufacturers, 20
Android project, 20
device convergence, 17
first Android phone, 20
first cell phone, 13
first-generation, 13
form factors, 15
functionality, 11–13
Google Internet model, 19
Open Handset Alliance (OHA), 20
operators, 21
platform market penetration, 18–19
proprietary platforms, 17
time-waster games, 14
WAP, 15–16
limitations, 340
loss of signal, 367
personal, testing, 330
preproduction, 368
profiles, 405–407
RAM size, 405
required features, specifying, 129–130

screen
captures, 435
sizes, configuring, 130, 301
types, 305-306
settings, accessing, 292
target
acquiring, 335
availability, 334-335
identifying, 333
selection mode, 66
testing, 336
third-party firmware modifications, 365
tracking, 332
unexpected configuration changes, testing, 373
upgrades, 372
USB connections, 409
diagnostics
anonymous, gathering, 354
code, 358-359
leveraging, 353
Dialog class, 252
reference website, 260
website, 321
DialogFragment class, 238, 250, 260
dialogs
adding
activities, 254
fragment-based method, 252
legacy method, 251
alerts, 252
basic, 252
character pickers, 252
date pickers, 253
fragment method, 257-260
legacy method, 253
customizing, 256
defining, 254
destroying, 255-256
dismissing, 255
initializing, 254
launching, 255
lifecycle, 254
progress, 253
time pickers, 253
types, 252-253
websites, 260
dictionary, 291
digital clocks, 196
DigitalClock control, 196
<dimen> tag, 152
dimensions, 152
accessing, 153
defining, 152
resource file example, 152
screens, 312
storing, 139
unit measurements, 152
directories
accessing, 105
applications, 278
cache files, 282-283
default application
reading files, 280
writing files, 279
files, listing, 278, 282
important, 432
qualifiers, 309-313
resources, 138
subdirectories
creating, 282
retrieving, 279

disadvantages, 28
dismissDialog( ) method, 254
dismissing dialogs, 255
displaying
  AVDs, 404-405
  data to users
    adjusting progress, 194
    clocks, 196-197
    progress bars, 192-193
    ratings, 194-195
    time passage, 195-196
  Eclipse windows side by side, 440
  network status, 418
  TextView control, 173-174
DisplayMetrics class, 304
distribution, 25, 343-344, 385. *See also* publication
  ad revenue, 379
  alternatives, 393-394
  Android Market, 385, 392
    contact/consent options, 390
    developer accounts, creating, 385-387
    fees, 390
    help, 390
    listing details, 388
    marketing assets, uploading, 388
    publishing options, 390
    removing applications, 393
    return policies, 392
    upgrading applications, 392
    uploading applications, 387
    website, 385
  choosing, 377-378
  in-application billing, 379
  intellectual property protection, 378
  market reviews, 344
  options, 27-28
  self-distribution, 394-395
  statistics collection, 379
dmtracedump tool, 98
dock mode, 313
Document Object Model Core package, 45
documentation, 83-85
  maintenance/support, 345
  mobile development projects, 337-338
  SDK, 43
  sections, 83
  website, 83
<drawable> tag, 153-154
drawables
  accessing, 154
  defining, 153
  ShapeDrawable class, 153
  storing, 139

E

Eclipse, 38
  ADT plug-in, 46-47, 88
    creating resources, 143-146
    resource editors, 89
    layout resource editor, 164-166
    UI designer, 90
  applications, signing, 383
  AVDs, creating, 403-404
  code
    autocomplete, 443
    build errors, 447
    classes, creating, 443
    comments, 446
    formatting, 444

imports, 443
methods, creating, 443
QuickFix feature, 446
refactoring, 444–445
reorganizing, 446
DDMS perspective, 87
debug configurations, creating, 409
device USB connections, 409
emulator GPS location, creating, 412
imported classes, 73
layouts, designing, 164–166
log filters, creating, 441
manifest files
application-wide settings, 121
editing, 120
instrumentation settings, 122
package-wide settings, 121
permissions, 121
perspectives, 47, 440
projects, creating, 62–64
application information, 64
build targets, 63
file locations, choosing, 63
launch activities, 64
minimum SDK version, 64
names, 62
package names, 64
Rename tool, 444
resources, creating, 143–146
run configurations, creating, 409
sample projects, adding, 52
searches, 442
Snake application
adding, 54
AVD, creating, 56–57
launch configuration, creating, 58–59
running, 59–60
source control services, 439
stepping through code, 72
tabs, closing, 441
tasks, 442
two sections of same file, viewing, 441
website, 52
windows
maximizing, 440
minimizing, 440
open, limiting, 441
viewing side by side, 440

edit( ) method, 265

editing
content providers, records
adding, 297–298
deleting, 298–299
updating, 298
manifest file, 120
application-wide settings, 121
Eclipse, 120
instrumentation settings, 122
manually, 123–124
package-wide settings, 121
permissions, 121
preferences, 266–267
wallpaper, 419

EditText class, 176-178
defining, 177
hints, 177
InputFilter interface, 178–179
lines, 177

EditTextPreference class, 274

elapsedRealtime( ) method, 196
ellipsis (…), 174
ellipsize attribute, 174
ems, 174
emulator, 49
AVDs
AVD Manager, 402
creating, 403-404
display options, 404-405
hardware configuration settings, 405-407
settings, 401
benefits, 368
best practices, 399-401
calling between, 413
console
commands, 419
connecting, 416
GPS coordinates, sending, 418
incoming calls, simulating, 416
network status, displaying, 418
power settings, 418
SMS messages, simulating, 416
Control pane, 434
disadvantages, 369
files, managing, 432-433
GPS location, 411-413
Hierarchy Viewer, launching, 92
icon tips, 419
launching, 60
AVD Manager, 411
options, 407
specific projects, 408-409
limitations, 420-421
location, configuring, 77
messaging between, 415
MyFirstAndroidApp
debugging, 69-73
running, 67-68
overview, 85, 399
pages, adding, 419
PeakBagger 1.0, 86
performance, 407-408
screen captures, 435
smartphone-style example, 49
Snake app, running, 59-60
startup options, 408
tablet-style example, 50
tips, 419
unlocking, 60
wallpaper, editing, 419
website, 86, 100, 421
End User License Agreement (EULA), 351
enforcing
permissions, 134
platform requirements, 129-130
device features, 129-130
input methods, 129
screen sizes, 130
system requirements, 125-128
entries attribute, 183
environment
configuring
hardware, 39-41
installation, 38
operating systems, 39
software requirements, 37-38
SDK, upgrading, 41

testing, 365
clean states, 366-367
device configurations, 365-366
real world simulations, 367
testing with Snake application, 53
adding to Eclipse, 54
AVD, creating, 56-57
launch configuration, creating, 58-59
running in emulator, 59-60
Environment class, 284
etc1tool tool, 99
EULA (End User License Agreement), 351
Exerciser Monkey tool, 373, 376
expansion beyond smartphones, 23
extensibility, 341-342
external activities, launching, 114
external file storage, 283-284
external libraries, 130
Extract Local Variable tool, 445
Extract Method tool, 445
extreme programming website, 346

F

feasibility
assessments, 336
testing, 356
FieldNoteListFragment class, 240-243
FieldNoteViewActivity class, 246
FieldNoteWebViewFragment class, 243-244
FierceDeveloper newsletter website, 7
File class, 282-284
File Explorer, 430
browsing, 432
copying files, 432-433
deleting files, 433
drag-and-drop operations, 433
File menu commands
New, Android Project, 62
New, Other, 54
fileList( ) method, 278
files. *See also* directories
accessing, 105, 278
AndroidManifest.xml, 65
browsing, 432
cache
storing, 282-283
subdirectory, retrieving, 279
copying, 432-433
core, 65
creating, 282
default application directories, 279-280
deleting, 278, 433
files subdirectory, 279
gen/com/androidbook/myfirstandroidapp/R.java, 65
handles, retrieving, 278
HPROF, creating, 429
important directories, 432
listing, 278, 282
managing
best practices, 276-277
methods, 278-279
manifest
activities, registering, 131-133
applications, 125, 134
broadcast receivers, registering, 133
content providers, registering, 133
editing, 89, 120-124

external libraries, 130
features, 135
GL texture compression formats, 131
names, 119
overview, 119-120
package names, 124
permissions, 133-135
platform requirements, enforcing, 129-130
publication preparations, 381
reference website, 136
screen compatibility, 131
SDK versions, targeting, 126-128
services, registering, 133
system requirements, enforcing, 125-128
opening, 279
permissions, 277
private, 277
proguard.cfg, 65
project.properties, 65
raw, 160
accessing, 161
defining, 160
reading byte-by-byte, 280
storing, 140
readable, 277
reading
byte-by-byte, 280
XML files, 280-281
res/drawable, 65
res/layout/main.xml, 65
res/values/strings.xml, 65
resources
accessing, 142
aliases, 162
alternative, 141-142. *See also* resources, alternative
animations, 139, 156
application, retrieving, 105
Boolean, 139, 149-150
colors, 139, 151
creating, 143-146
default, 141-142, 308
defined, 137
dimensions, 139, 152-153
directories, 138
drawables, 139, 153-154
frame-by-frame animations, 156-157
images, 139-141, 154-155
integers, 139, 150-151
layouts. *See* resources, layouts
menus, 139, 158-159
mixed-type arrays, 139
primitive, 140
raw files, 140, 160-161
references, 161-162
selectors, 156
storing, 137-140
strings, 139, 146-149
styles, storing, 140
system, 167-168
themes, storing, 140
tweened animations, 157-158
types, 138-140
user preferences, 269-272
websites, 168
XML files, 139, 159-160

source control services, 439
storing, 141, 283-284
subdirectory, 279
writeable, 277
XML, 159
accessing, 160
defining, 160
parsing utilities, 280
reading, 280-281
storing, 139

filter( ) method, 179

filters
Android Market, 381
input, 178-179
intents, 114
logging, 441
market, 395

firmware upgrades, testing, 345

first Android handset, 20

first cell phone, 13

first-generation mobile phones, 13

Flash support, 33

folders, 65

footers, 226

forceError( ) method, 69

Foreground attribute, 208

foregroundGravity attribute, 208

form factors (phones), 15

formats
code, 444
colors, 151
images, 154
resource references, 161
strings, 146-147

forms
autocompletion, 179-181
LinearLayoutview, 209-210

forums, 7

Fragment class
callback methods, 237
defined, 104
reference website, 250
subclasses, 237
website, 118

<fragment> tag, 235

fragments
Activity classes, creating, 246-247
applications, designing, 238
attaching, 237
compatibility, 305
creating, 237
defined, 104, 111
defining, 235
destroying, 237
detaching, 237
dialogs, 238, 257-260
layout resource files, creating, 244-246
legacy support, 247-248
lifecycle, 234-235
ListFragment, implementing, 240-243
lists, 238
MP3 music player example, 111-113
overview, 233
pausing, 237
resuming activity, 237
screen workflow, 234
specialty, 237
stopping activity, 237

updates, 236
user preferences, 238
visibility, 237
web content, 238
website, 118
WebView, implementing, 243-244
workflow flexibility, improving, 111-113
frame-by-frame animations, 156-157
FrameLayout class, 207
attributes, 207-209
website, 231
framework
distribution, 25
important packages, 43
third-party APIs, 45
free development, 23
free platform, 24

G

galleries, 205, 222
Gallery class, 231
GalleryView class, 222
game application activities, 106
garbage collection, 428
gen/com/androidbook/myfirstandroidapp/R.java file, 65
getAll( ) method, 265
getApplicationContext( ) method, 104
getAssets( ) method, 105
getBoolean( ) method, 150, 265
getCacheDir( ) method, 279, 282
getColor( ) method, 151
getContentResolver( ) method, 298
getDimension( ) method, 153
getDir( ) method, 279
getExternalCacheDir( ) method, 279, 283-284
getExternalFilesDir( ) method, 279, 284
getExternalStoragePublicDirectory( ) method, 284
getExternalStorageState( ) method, 284
getFilesDir( ) method, 279
getFileStreamPath( ) method, 279
getFloat( ) method, 265
getInt( ) method, 265
getLocation( ) method, 78
getLong( ) method, 265
getResources( ) method, 105
getSharedPreferences( ) method, 105
getString( ) method, 266
getStringSet( ) method, 266
getText( ) method, 173, 178
GL texture compression formats, configuring, 131
GNU General Public License Version 2 (GPLv2), 25
Google
Analytics, 45, 361
Android project, 20
APIs, 45
application development, 22
common intents, 115
Developers
blog, 350
Guide website, 52, 100
Experience devices, 335
Internet model, 19
Maps, 413
Billing SDK, 46

Market Licensing, 46
Open Handset Alliance (OHA), 20
manufacturers, 20
mobile operators, 21
website, 35
Team Android Apps website, 7
TV compatibility, 319–321
GPS support, 406
coordinates
sending, 418
website, 413
emulator location, 77, 411
last known location, 77
MyFirstAndroidApp, 76–78
GPU emulation, 406
graphics, 154
accessing, 155
buttons, 185
drawables
accessing, 154
defining, 153
ShapeDrawable class, 153
formats, 154
GL texture compression formats, configuring, 131
Nine-Patch Stretchable Graphics, 95, 100
accessing, 155
creating, 95–97, 155
defined, 95, 155
device compatibility, 306
scaling, 95
storing, 139–141
screen captures, 435
gravity attribute
GridLayout class, 218
LinearLayout class, 210
RelativeLayout class, 212
green robot, 24
GridLayout class, 216-220, 231
grids, 222
GridView class, 222, 231
groups
permissions, 135
radio buttons, 187–189
growing platform, 28
GSM modem support, 406

H

Handango, 393
handles (files), retrieving, 278
handling
click events, 224
runtime changes, 318, 321
single device bugs, 359
hardware
application interaction, 34
availability, 334–335
bugs, handling, 359
clean states, 366–367
compatibility, 301
alternative resources. *See* compatibility, alternative resources
best practices, 322
densities, 301
fragments, 305
Google TVs, 319–321
maximizing, 303

nine-patch stretchable graphics, 306
OpenGL versions, 301
platforms, 301
runtime changes, handling, 321
screen sizes, 301
screen types, 305-306
Support (Compatibility) Package, 305
tablets, 318-319
user interfaces, 303-304
working squares, 306-307
competitive, 23
convergence, 17
databases, 330
data, storing, 332
devices for tracking, choosing, 332
functions, 332
third-party, 333
debugging
configuring, 39-41
enabling, 39-41
MyFirstAndroidApp, 78-79
operating systems, configuring, 39
files
browsing, 432
copying, 432-433
deleting, 433
storing, 332
fragmentation, 365-366
Google Experience, 335
history
Android manufacturers, 20
Android project, 20
device convergence, 17
first Android phone, 20
first cell phone, 13
first-generation, 13
form factors, 15
functionality, 11-13
Google Internet model, 19
Open Handset Alliance (OHA), 20
operators, 21
platform market penetration, 18-19
proprietary platforms, 17
time-waster games, 14
WAP, 15-16
limitations, 340
loss of signal, 367
personal, testing, 330
preproduction, 368
profiles, 405-407
RAM size, 405
required features, specifying, 129-130
screen
captures, 435
sizes, configuring, 130, 301
types, 305-306
settings, accessing, 292
target
acquiring, 335
availability, 334-335
identifying, 333
selection mode, 66
testing, 336
third-party firmware modifications, 365
tracking, 332
unexpected configuration changes, testing, 373
upgrades, 372
USB connections, 409

hash symbols (#), 151
headers, 226
heap statistics, monitoring, 427
height
  input boxes, 177
  text, 174
Hierarchy View perspective (Eclipse), 47
Hierarchy Viewer
  launching, 92
  layouts, inspecting, 220
  modes, 92
    layout controls above application content, 92
    Layout View, 92–93
    Pixel Perfect, 94
    switching, 92
  online tutorial, 94
  overview, 91
  user interface optimization, 94
hint attribute (EditText control), 177
history of mobile devices
  Android manufacturers, 20
  Android project, 20
  device convergence, 17
  first Android phone, 20
  first cell phone, 13
  first-generation, 13
  form factors, 15
  functionality, 11–13
  Google Internet model, 19
  Open Handset Alliance (OHA), 20
    manufacturers, 20
    mobile operators, 21
    website, 35
  operators, 21
  platform market penetration, 18–19
  proprietary platforms, 17
  time-waster games, 14
  WAP, 15–16
Home screen tips, 419
Home section (documentation), 83
HoneycombGallery sample app, 51
horizontal progress indicators, 192
HorizontalScrollView class, 229
hprof-conv tool, 98
HPROF files, creating, 429
HTTP support package, 45
hybrid development method, 329

I

<i> tag, 147
icons
  alternative resources, 310
  applications, configuring, 125, 380
  emulator tips, 419
ICS (Ice Cream Sandwich), 5
id fields, 223
IDEs (integrated development environments), 25. *See also* Eclipse
ImageButton class, 185
images, 154
  accessing, 155
  buttons, 185
  drawables
    accessing, 154
    defining, 153
    ShapeDrawable class, 153
  formats, 154
  GL texture compression formats, configuring, 131

Nine-Patch Stretchable Graphics, 95, 100
accessing, 155
creating, 95-97, 155
defined, 95, 155
device compatibility, 306
scaling, 95
storing, 139-141
screen captures, 435
ImageSwitcher controls, 230
importing
classes (Eclipse), 73
code, 443
improving workflow flexibility, 111-113
in-application billing, 379
inches, 152
incoming calls, simulating, 416, 434
indeterminate progress indicators, 193
indicators
clocks
analog, 197
digital, 196
progress bars, 192-193
adjusting, 194
circular, 192
horizontal, 192
indeterminate, 193
title bars, adding, 193
ratings, 194-195
time passage, 195-196
initializing
activity data, 108-109
dialogs, 254
input, 176
autocompletion, 179-181
buttons, 183
basic, 183-184
check boxes, 183, 186
image buttons, 185
radio, 183, 187-189
switches, 183, 187
toggle, 183, 186
dates, 190-191
filters, 178-179
hints, 177
input box height, 177
retrieving, 176-178
specifying, 129
spinners, 181-183
text
alternative resource qualifiers, 315
contextual links, 174-176
displaying, 173-174
ellipsis, 174
height, 174
Toast messages, 185
width, 174
times, 191
InputFilter interface, 178-179
inspecting
layouts, 220
user interfaces at pixel level, 94
installing
development environment, 38
instructions website, 39
testing, 372
Instrumentation tab (Eclipse), 122

<integer> tag, 150

integers, 150
- accessing, 150
- creating, 151
- defining, 150
- storing, 139

integrated development environments (IDEs), 25. *See also* Eclipse

integration
- applications, 26
- points, testing, 371

intellectual property protection, 378

Intent class, 104

<intent-filter> tag, 132

intents, 104
- action/data, 114
- additional information, adding, 115
- application navigation, 115–116
- broadcasting, 117
- filters, 114, 132–133
- Google common, 115
- launching activities
  - by class name, 113
  - external, 114
- primary entry points, choosing, 132
- website, 118

interfaces
- InputFilter, 178–179
- OnChronometerTickListener, 196
- SharedPreferences, 264
  - methods, 265
  - reference website, 273
- SharedPreferences.Editor
  - methods, 266
  - reference website, 273
- TabContentFactory, 226

internationalization, testing, 372

INTERNET permission, 75

interoperability (applications), 342

isFinishing( ) method, 111

ISO 3166-1-alpha-2 Regions website, 322

ISO 639-1 Language codes website, 322

Issue Tracker website, 41

italic strings, 147

iterative development, 327, 346

J

Java
- advantages, 26
- autocomplete, 443
- build errors, 447
- classes, creating, 443
- comments, 446
- formatting, 444
- imports, 443
- methods, creating, 443
- packages, 44, 284
- perspective (Eclipse), 47
- QuickFix feature, 446
- refactoring code, 444–445
- reorganizing, 446
- website, 52

Javadoc comments, 446

Java Development Kit (JDK), 37

JSON (JavaScript Object Notation) package, 45

junit package, 44

K

key attribute, 269

keyboard
- alternative resource qualifier, 314
- support, 406

killer application chances, increasing, 374

killing activities, 109-110

L

landscape-mode alternative picture graphic, 316

languages
- alternative resource qualifier, 312
- ISO 639-1 codes website, 322

last known location, finding, 77

launch activities, creating, 64

launch configurations
- creating
  - MyFirstAndroidApp, 66-67
  - Snake application, 58-59
- Debug, 66
- Run, 66

launching
- activities by class name, 113
- applications, 114
- AVD Manager, 47
- dialogs, 255
- emulator, 60
  - AVD Manager, 411
  - options, 407
  - specific projects, 408-409
- Hierarchy Viewer, 92
- SDK Manager, 47

layout_gravity attribute, 208

Layout View mode (Hierarchy Viewer), 92-93

layoutopt tool, 220, 375

LayoutParams class, 205

layouts
- attributes, 205-206
- controls, 172
- creating
  - programmatically, 201-203
  - XML, 199-200
- declaring website, 231
- Eclipse layout resource editor, 164-166
- frames, 207-209
- grids, 216-220
- inspecting, 220
- linear, 206, 209-210
- multiple, 220
- nesting, 207
- relative, 211-214
- resource files
  - accessing, 166
  - alternative versions, 167
  - background color/text example, 162-163
  - defined, 162
  - fragments, 244-246
- storing, 140
- tables, 214-216
- types, 205

legacy support, 6
- contacts, 293-294
- fragments, 247-248
- package, 45
- platform, 248

libraries, 130

licensing, 25
- Agreement website, 52
- End User License Agreement (EULA), 351
- Google Market Licensing, 46
- LVL, 378
- SDK license agreement, 42-43

lifecycles
- activities, 106–107
- applications, 34
- dialogs, 254
- fragments, 235

LinearLayout class, 172, 209-210, 231

lines attribute
- EditText control, 177
- TextView control, 174

links, creating, 174-176

Linux
- configuration website, 39
- operating system, 30

ListActivity class, 225, 231

listening preferences, 267

ListFragment class, 238
- implementing, 240–243
- reference website, 250

listing details (Android Market), 388

ListPreference class, 274

lists, 222
- files, 278, 282
- fragments, 238
- user interfaces, creating, 225

ListView class, 222, 231

live servers, managing, 345

local variables, extracting, 445

location-based services, adding
- coordinates
  - sending, 418
  - website, 413
- emulator location, 77
- last known location, 77
- MyFirstAndroidApp, 76–78

LocationManager class, 81

Log class
- methods, 73
- website, 81

logcat tool, 86-87, 99, 374, 436-437

logging
- adding, 73–74
- filters, creating, 441
- turning off, 381

loss of signal testing, 367

low-risk porting, 345

lowest common denominator development method, 327-328

LunarLander sample app, 51

LVL (License Verification Library), 378

M

maintenance
- application diagnostics, leveraging, 353
- documentation, 338
- mobile development, 344–345

managedQuery( ) method, 224, 286
- Browser class, 290
- CallLog class, 289
- MediaStore.Audio.Media class, 288

managers
- AVD
  - launching, 47
  - overview, 48
- SDK
  - launching, 47
  - overview, 48

managing
- activity transitions
  - action/data, 114
  - additional information, adding, 115

application navigation, 115-116
broadcasting, 117
filters, 114, 132-133
Google common, 115
launching activities, 113-114
primary entry points, choosing, 132
website, 118
Android Market applications
removing, 393
return policies, 392
upgrades, 392
AVDs, 402
device databases, 330
data, storing, 332
devices for tracking, choosing, 332
functions, 332
third-party databases, 333
files
best practices, 276-277
methods, 278-279
live servers, 345
memory allocations, 430
testing environment, 365
clean states, 366-367
device configurations, 365-366
real world simulations, 367

manifest files
activities, registering, 131
intent filters, 132-133
primary entry points, 132
applications
enforced permissions, registering, 134
icons, 125
names, 125
versioning, 125
broadcast receivers, registering, 133
content providers, registering, 133
editing, 89, 120
application-wide settings, 121
Eclipse, 120
instrumentation settings, 122
manually, 123-124
package-wide settings, 121
permissions, 121
external libraries, 130
features, 135
GL texture compression formats, 131
names, 119
overview, 119-120
package names, 124
permissions
application enforced, registering, 134
groups, 135
protection levels, 135
required, registering, 133-134
platform requirements, enforcing, 129-130
device features, 129-130
input methods, 129
screen sizes, 130
publication preparations, 381
reference website, 136
screen compatibility, 131
SDK versions, targeting, 126-128
maximum, 128
minimum, 127
specifying, 128
services, registering, 133
system requirements, enforcing, 125-128

Manifest tab (Eclipse), 121
manufacturers, 20, 334
map coordinates website, 413
MarginLayoutParams class, 205
Market (Android), 385, 392
  alternatives, 393-394
  applications
    removing, 393
    upgrading, 392
    uploading, 387
  contact/consent options, 390
  developer accounts, creating, 385-387
  fees, 390
  filters, 381, 395
  help, 390
  listing details, 388
  marketing assets, uploading, 388
  packaging requirements, 381
  publishing options, 390
  return policies, 392
  reviews, 344
  website, 7, 385, 395
marketing. *See* distribution
markets
  focus, 23
  targeting, 344
mascot, 24
max VM application heap size, 407
maximizing, 367
  application
    popularity, 374
    upgrades, 371
    usability, 370
  automation, 368
  backup services, 373
  billing, 373
  black-box testing, 369
  build validations, 368
  compatibility, 303
  conformance, 372
  device upgrades, 372
  emulators
    benefits, 368
    disadvantages, 369
  installations, 372
  integration points, 371
  internationalization, 372
  performance, 373
  preproduction devices, 368
  remote servers/services, 370
  third-party standards, 371
  unexpected configuration changes, 373
  visual appeal, 370
  white-box testing, 369
  windows (Eclipse), 440
maxLines attribute, 174
measureAllChildren attribute, 208
measurements
  dimensions, 152
  ems, 174
media
  accessing, 286-288
  adding, 74-76
  audio file titles, retrieving, 287
  queries, 288
MediaPlayer class
  methods, 74
  website, 81
MediaStore content provider, 286-288
  audio file titles, retrieving, 287
  classes, 286-287
  website, 300

memory allocations, 430

menus, 158

accessing, 159

creating, 159

defining, 158

storing, 139

messages

between emulators, 415

Toast, 185

methods

addFooterView( ), 226

addHeaderView( ), 226

addPreferencesFromResource( ), 271

addTab( ), 227-229

addToBackStack( ), 236

addView( ), 204

addWord( ), 291

apply( ), 267

clear( ), 266

clearCheck( ), 189

commit( ), 266

contains( ), 265

create( ), 75

createTabContent( ), 228

creating, 443

delete( ), 299

deprecated, 252

dismissDialog( ), 254

edit( ), 265

elapsedRealtime( ), 196

file management, 278-279

filter( ), 179

forceError( ), 69

Fragment class, 237

getAll( ), 265

getApplicationContext( ), 104

getAssets( ), 105

getBoolean( ), 150, 265

getCacheDir( ), 282

getColor( ), 151

getContentResolver( ), 298

getDimension( ), 153

getExternalCacheDir( ), 283-284

getExternalFilesDir( ), 284

getExternalStoragePublicDirectory( ), 284

getExternalStorageState( ), 284

getFloat( ), 265

getInt( ), 265

getLocation( ), 78

getLong( ), 265

getResources( ), 105

getSharedPreferences( ), 105

getString( ), 266

getStringSet( ), 266

getText( ), 173, 178

isFinishing( ), 111

Log class, 73

managedQuery( ), 224, 286

Browser class, 290

CallLog class, 289

MediaStore.Audio.Media class, 288

MediaPlayer class, 74

obtainTypedArray( ), 162

onActivityCreated( ), 237

onAttach( ), 237

onCheckedChangedListener( ), 189

onClick( ), 185

onCreate( ), 108, 237

onCreateDialog( ), 254

onCreateView( ), 237

onDestroy( ), 110

onDestroyView( ), 237
onDetach( ), 237
onItemClick( ), 225
onListItemClick( ), 225
onPause( )
  Activity class, 109
  fragments, 237
onPrepareDialog( ), 254
onPressChanged( ), 194
onRatingChanged( ), 195
onResume( )
  Activity class, 109
  fragments, 237
onRetainNonConfigurationInstance( ), 318
onSaveInstanceState( ), 110
onStart( ), 237
onStop( ), 76, 237
openFileOutput( ), 279
parse( ), 75
playMusicFromWeb( ), 75
putBoolean( ), 266
putFloat( ), 266
putInt( ), 266
putLong( ), 266
putString( ), 266
putStringSet( ), 266
registerOnSharedPreferenceChange-Listener( ), 267
release( ), 75
remove( ), 266
removeDialog( ), 254
set, 229
setAdapter( ), 224
setContentView( ), 173
setFilters( ), 179
setIndicator( ), 229
setListAdapter( ), 225
setOnClickListener( ), 185
setOnItemClickListener( ), 224
setOnTimeChangedListener( ), 191
setText( ), 173, 178
setup( ), 229
SharedPreferences interface, 265
SharedPreferences.Editor, 266
showDialog( ), 254
start( )
  Chronometer control, 196
  MediaPlayer class, 75
stop( )
  Chronometer control, 196
  MediaPlayer class, 75
unregisterOnSharedPreferenceChange-Listener( ), 267
Uri class, 74

millimeters, 152

minimizing windows (Eclipse), 440

minLines attribute, 174

mistakes
  avoiding, 355, 360
  testing, 375

mixed-type arrays, storing, 139

mksdcard tool, 99

mnt/sdcard/ directory, 432

mnt/sdcard/download/ directory, 432

mobile country code alternative resource qualifier, 312

mobile development, 325
  applications
    feasibility assessments, 336
    interoperability, 342

best practices, 355
    code diagnostics, 358-359
    code reviews, 358
    coding standards, 357
    feasibility, testing, 356
    mistakes, avoiding, 360
    reference websites, 361
    single device bugs, 359
    software process, choosing, 356
    tools, leveraging, 360
billing/revenue generation, 351-352
    in-app, 379
    testing, 373
challenges, 325
devices
    availability, 334-335
    databases, 330-333
    limitations, 340
diagnostics
    anonymous, gathering, 354
    code, 358-359
    leveraging, 353
distribution, 25, 343-344, 385. *See also* publication
    ad revenue, 379
    alternatives, 393-394
    Android Market, 385-393
    choosing, 377-378
    in-application billing, 379
    intellectual property protection, 378
    market reviews, 344
    options, 27-28
    self-distribution, 394-395
    statistics collection, 379
documentation, 337-338
extensibility, 341-342
implementation steps, 342
iteration, 327
maintenance/support, 344-345
manufacturers/operators risks, 334
mistakes, avoiding, 355
network-driven applications, 340
outside influences, 335
personal device testing, 330
quality assurance, 336-338
requirements, 327-330
risk assessment, 333
rules, 347-348
security, 351
source control systems, choosing, 339
stability/responsiveness, 349-350
standalone applications, 340
target devices, 335
testing, 343
third-party standards, 352
tools, 354
updates/upgrades, 354
user demands, meeting, 348
user interfaces, 348-349
versioning schemes, choosing, 339-340
waterfall approaches, 326
mobile network code alternative resource qualifier, 312
mobile operators, 21
Mobiletuts+ website, 7
monkey tool, 99
monkeyrunner tool, 99
MoreAsserts class, 359
Motorola DynaTAC 8000X, 13
MP3 playback, adding, 75

MultiAutoCompleteTextView control, 181

multiple layouts, 220

multiple screens, 136, 322

MyFirstAndroidApp

- AVD, creating, 65
- core files, 65
- debugging
  - emulator, 69-73
  - devices, 78-79
- launch configuration, 66-67
- location-based services, 76-78
- logging support, 73-74
- media support, 74-76
- new project, creating, 62-64
  - application information, 64
  - build targets, 63
  - file locations, choosing, 63
  - launch activities, 64
  - minimum SDK version, 64
  - names, 62
  - package names, 64
- running, 67-68

N

names

- applications, configuring, 125, 380
- manifest file, 119
- packages, 64, 124
- projects, 62
- SDKs, 25

native versus third-party applications, 26, 33

navigation

- applications, 115-116
- methods alternative resource qualifier, 315

nesting, 207

network-driven applications, 340

network status, displaying, 418

New, Android Project command (File menu), 62

New, Other command (File menu), 54

night mode, 313

Nine-Patch Stretchable Graphics, 95

- accessing, 155
- creating, 95-97, 155
- defined, 95, 155
- device compatibility, 306
- scaling, 95
- website, 100

normal protection level (permissions), 135

NotePad sample app, 51

O

obtainTypedArray( ) method, 162

Official Android Developers Blog, 35

OHA (Open Handset Alliance), 20

- manufacturers, 20
- mobile operators, 21
- website, 35

onActivityCreated( ) method, 237

onAttach( ) method, 237

onCheckedChangedListener( ) method, 189

OnChronometerTickListener interface, 196

onClick( ) method, 185

onCreate( ) method, 108, 237

onCreateDialog( ) method, 254

onCreateView( ) method, 237

onDestroy( ) method, 110

onDestroyView( ) method, 237

onDetach( ) method, 237

onItemClick( ) method, 225
OnItemClickListener class, 224
onListItemClick( ) method, 225
onPause( ) method
  Activity class, 109
  fragments, 237
onPrepareDialog( ) method, 254
onProgressChanged( ) method, 194
onRatingChanged( ) method, 195
onResume( ) method
  Activity class, 109
  fragments, 237
onRetainNonConfigurationInstance( ) method, 318
onSaveInstanceState( ) method, 110
onStart( ) method, 237
onStop( ) method, 76, 237
Open Handset Alliance. *See* OHA
open platform, 24
open-source platforms, 25
openFileInput( ) method, 279
openFileOutput( ) method, 279
OpenGL compatibility, 301
opening files, 279
operating systems
  application interaction, 34
  configuring for device debugging, 39
  supported, 25
operators, 334
optimizing user interfaces, 94
org.apache.http package, 45
org.json package, 45
org.w3c.dom package, 45
org.xml.sax package, 45
org.xmlpull package, 45
organizing activities (fragments)
  Activity classes, creating, 246-247
  applications, designing, 238
  attaching, 237
  compatibility, 305
  creating, 237
  defined, 104, 111
  defining, 235
  destroying, 237
  detaching, 237
  dialogs, 238, 257-260
  layout resource files, creating, 244-246
  legacy support, 247-248
  lifecycle, 235
  ListFragment, implementing, 240-243
  lists, 238
  MP3 music player example, 111-113
  overview, 233
  pausing, 237
  resuming activity, 237
  screen workflow, 234
  specialty, 237
  stopping activity, 237
  updates, 236
  user preferences, 238
  visibility, 237
  web content, 238
  website, 118
  WebView, implementing, 243-244
  workflow flexibility, improving, 111-113
orientation, screens
  alternative resources, 316
  directory qualifier, 313
orientation attribute
  GridLayout class, 217
  LinearLayout class, 210

P

packages
- android, 44
- C2DM, 45
- commonly used, 34
- dalvik, 44
- Google
  - Analytics SDK, 45
  - APIs, 45
  - Market Billing, 46
  - Market Licensing, 46
- important, 43
- java, 44
- java.io, 284
- javax, 44
- junit, 44
- legacy support, 45
- manifest file settings, 121
- names, 64, 124
- org.apache.http, 45
- org.json, 45
- org.w3c.dom, 45
- org.xml.sax, 45
- org.xmlpull, 45
- provider, 285, 300
- Support (Compatibility)
  - adding, 248-249, 305
  - website, 118
- test, 359
- third-party APIs, 45
- view, 171
- widget, 171

packaging
- code preparations, 380
  - Android Market, 381
  - application names/icons, 380
  - debugging/logging, turning off, 381
  - manifest file, 381
  - permissions, 382
  - target platforms, 381
  - versioning, 380
- publication requirements, 380
- release versions, testing, 384
- signing, 382-384

pages, adding, 419

ParisView project, 165

parse( ) method (Uri class), 75

PeakBagger 1.0, 86

performance
- alternative resources, 317
- emulator, 407-408
- testing, 373

PerformanceTestCase class, 359

<permission> tag, 135

permissions
- ad-hoc, 31
- adding, 75
- application specific, 31
- contacts, 292
- content providers, 289
- files, 277
- groups, 135
- INTERNET, 75
- manifest file settings, 121
- protection levels, 135
- registering, 133-134
- settings, 292
- verifying, 382
- voicemail, 291
- website, 136

Permissions tab (Eclipse), 121

personal devices, testing, 330

perspectives (Eclipse), 440, 47

Pixel Perfect mode (Hierarchy Viewer), 92-94

pixels, 152, 314

platform

advantage over competitors, 25

architecture, 29

Dalvik VM, 31

Linux operating system, 30

compatibility, 301

complete, 23

definition, 23

differences

application development, 26

application integration, 26

development tools, 25

distribution options, 27

growing platform, 28

IDEs, 25

Java, 26

open-source, 25

publication, 27

SDKs, 25

free, 24

legacy support, 248

market penetration, 18-19

open, 24

open-source, 25

proprietary, 17

requirements

device features, 129-130

enforcing, 129-130

input methods, 129

screen sizes, 130

security, 31

ad-hoc permissions, 31

application-specific permissions, 31

applications running as operating system users, 31

certificate signing, 32

developer registration, 32

target, 328, 381

playMusicFromWeb( ) method, 75

plug-ins, ADT Eclipse, 46-47, 88

resources

creating, 143-146

editors, 89

layout resource editor, 164-166

UI designer, 90

plural strings, 149

points, 152

popular applications, creating, 374

porting

documentation, 338

low-risk, 345

portrait-mode alternative picture graphic, 316

power settings, simulating, 418

pre-publication checklist website, 361

Preference class, 269

attributes, 269

reference website, 274

PreferenceActivity class, 268-273

PreferenceCategory class, 274

PreferenceFragment class, 238, 250

preferences

accessing, 105, 267-268

adding, 264

appropriateness, 263

data types supported, 264

editing, 266-267

overview, 263

PreferenceActivity class, 268

private, 264

reacting, 267
reading, 265
searching, 265
shared, 265
user
fragments, 238
loading up, 270-272
resource file, creating, 269-270
PreferenceScreen class, 274
<PreferenceScreen> tag, 269
preproduction devices, 368
primary entry points (activities), 132
primitive resources, storing, 140
private data, 277, 351
private preferences, 264
processes, stopping, 426
profiling user interfaces, 100
programming language choices, 32-33
progress
bars, 192-193
circular, 192
horizontal, 192
indeterminate, 193
title bars, adding, 193
dialogs, 253
ProgressBar class, 192-193
ProgressDialog class, 253, 260
proguard.cfg file, 65
project.properties file, 65
projects
application feasibility assessments, 336
creating, 62-64
application information, 64
build targets, 63
file locations, choosing, 63
launch activities, 64
minimum SDK version, 64
names, 62
package names, 64
device databases, 330
adapters, 223-224
data, storing, 332
devices for tracking, choosing, 332
functions, 332
third-party, 333
documentation, 337-338
manufacturers/operators, 334
outside influences, 334-335
ParisView, 165
quality assurance, 336-337
requirements, 327
customization, 328-329
hybrid approaches, 329
lowest common denominator, 327-328
third-parties, 330
use cases, creating, 329
risk assessment, 333
sample, 52
source control systems, 339
target devices
acquiring, 335
identifying, 333
versioning schemes, 339-340
prompt attribute, 183
proprietary platforms emergence, 17
protection levels (permissions), 135
protocols
HTTP support package, 45
WAP, 15-16
provider package, 285, 300
<provider> tag, 133

publication
ad revenue, 379
advantages, 27
alternatives, 393-394
Android Market, 385, 392
contact/consent options, 390
developer accounts, creating, 385-387
fees, 390
help, 390
listing details, 388
marketing assets, uploading, 388
publishing options, 390
removing applications, 393
return policies, 392
upgrading applications, 392
uploading applications, 387
website, 385
code preparation, 380
Android Market, 381
application names/icons, 380
debugging/logging, turning off, 381
manifest file, 381
permissions, 382
target platforms, 381
versioning, 380
distribution methods, 377-378, 385
in-application billing, 379
intellectual property protection, 378
pre-publication checklist website, 361
release versions, testing, 384
requirements, 380
self-publishing, 394-395
signing applications, 382-384
statistics collection, 379
putBoolean( ) method, 266
putFloat( ) method, 266
putInt( ) method, 266
putLong( ) method, 266
putString( ) method, 266
putStringSet( ) method, 266

Q

qualifiers (directory), alternative resources, 309
bad examples, 311
case, 310
combining, 310
customizing, 311
default resources, including, 311
good examples, 311
limits, 310
listing of, 311-313
requirements, 311
quality assurance
third-party standards, 352
risks, 336
clients/servers/cloud, 337
device tests, 336
real-world testing limitations, 337
test hardware, obtaining, 336
test plans, 338
querying
browsers, 290
call logs, 289
contacts
Contacts provider, 293-294
ContactsContract content provider, 295
quickly, 294-295
media, 288
QuickFix feature, 446

R

RAD (Rapid Application Development), 346
radio buttons, 187-189
RadioButton class, 183, 187-189
RadioGroup class, 187-189
RAM size (devices), 405
RatingBar class, 194-195
ratings, displaying, 194-195
raw files, 160
- accessing, 161
- defining, 160
- reading, 280
- storing, 140

reading
- files, 277
  - byte-by-byte, 280
  - default application directories, 280
  - XML files, 280-281
- preferences, 265

real world testing simulations, 367
records (content providers)
- adding, 297-298
- deleting, 298-299
- updating, 298

refactoring code, 444-445
Reference section (documentation), 84
referencing resources, 161
- aliases, 162
- format, 161
- string example, 161

region code alternative resource qualifier, 312
registering
- activities, 131
  - intent filters, 132-133
  - primary entry points, 132
- broadcast receivers, 133
- content providers, 133
- permissions, 133-134
- services, 133

registerOnSharedPreferenceChange Listener( ) method, 267
RelativeLayout class, 211-214, 231
release versions, testing, 384
release( ) method, 75
releasing activity data, 109
remove( ) method, 266
removeDialog( ) method, 254
Rename tool, 444
reorganizing code, 446
requirements
- Android Market packaging, 381
- permissions, 133-134
- platform
  - device features, 129-130
  - enforcing, 129-130
  - input methods, 129
  - screen sizes, 130
- projects, 327
  - customization, 328-329
  - hybrid approaches, 329
  - lowest common denominator, 327-328
  - third-parties, 330
  - use cases, creating, 329
- publication, 380
- software, 37-38
- system, 125-128

res folder, 65
res/drawable file, 65
res/layout/main.xml file, 65
res/values/strings.xml file, 65

Resource editor, 89
layouts, designing, 164-166
manifest file, 89
resources
accessing, 142
aliases, 162
alternative, 141-142
benefits, 308
creating, 309
data, saving, 318
defined, 308
directory qualifiers, 309-313
icons example, 310
organization schemes, 317
performance issues, 317
requesting, 316
runtime changes, handling, 318
screen orientations, 316
website, 321
animations
storing, 139
tweened, 156
types, 156
application, retrieving, 105
Boolean, 149
accessing, 150
defining, 150
storing, 139
colors, 151
accessing, 151
defining, 151
formats, 151
storing, 139
creating, 143-146
default, 141-142, 308
defined, 137
dimensions, 152
accessing, 153
defining, 152
resource file example, 152
storing, 139
unit measurements, 152
directories, 138
drawables
accessing, 154
defining, 153
ShapeDrawable class, 153
storing, 139
frame-by-frame animations, 156-157
images, 154
accessing, 155
formats, 154
nine-patch stretchable, 155
storing, 139-141
integers, 150
accessing, 150
arrays, 151
defining, 150
storing, 139
layouts
accessing, 166
alternative versions, 167
background color/text example, 162-163
creating, 199-200
defined, 162
Eclipse layout resource editor, 164-166
fragments, creating, 244-246
storing, 140

menus, 158
accessing, 159
creating, 159
defining, 158
storing, 139
mixed-type arrays, 139
primitive, 140
raw files, 160
accessing, 161
defining, 160
storing, 140
references, 161-162
selectors, 156
storing, 137-140
strings
accessing, 147-148
arrays, 149
bold/italic/underline, 147
formatting, 146-147
overview, 146
plural, 149
storing, 139
<string> tag, 146
styles, storing, 140
system, 167-168
themes, storing, 140
tweened animations
accessing, 158
defining, 157
overview, 157
spinning graphic example, 158
types, 138-140
user preferences
creating, 269-270
loading up, 270-272
websites, 168
XML files, 159
accessing, 160
defining, 160
storing, 139
Resources section (documentation), 84
responsiveness, 349-350, 361
retrieving
activity data, 109
application resources, 105
assets, 105
audio file titles, 287
cache subdirectory, 279
context, 104
file handles, 278
files subdirectory, 279
subdirectories, 279
return policies (Android Market), 392
revenue/billing generation, 351-352
in-app, 379
testing, 373
risk assessment, 333
application feasibility, 336
device availability, 334-335
manufacturers/operators, 334
quality assurance, 336-337
clients/servers/cloud, 337
device tests, 336
real-world testing limitations, 337
test hardware, obtaining, 336
target devices
acquiring, 335
identifying, 333
row attribute, 218
rowCount attribute, 217
rowSpan attribute, 218
rs:ResEnum application, 168

Rubin, Andy, 20

rules (design), 347-348

run configurations

- creating, 409
- MyFirstAndroidApp, 66

Run Configurations command (Run menu), 66

Run menu commands

- Debug Configurations, 58, 66
- Run Configurations, 66

running applications

- MyFirstAndroidApp, 66–68
- runtime changes, handling, 321
- Snake, 59–60

S

sample applications, 50-52

saving

- activities
  - data, 109
  - state, 110
- alternative resource data, 318

scale-independent pixels, 152

screens

- aspect ratio alternative resource qualifier, 313
- captures, 435
- compatibility
  - devices, 305–307
  - fragments, 305
  - mode, 306, 321
  - nine-patch stretchable graphics, 306
  - screen type support, 305–306
  - settings, 131
  - Support (Compatibility) Package, 305
  - user interfaces, 303–304
- density-independent pixels, 152
- dimensions alternative resource qualifier, 312
- information, accessing, 304
- multiple, 136, 322
- orientation, 313, 316
- pixel density alternative resource qualifier, 314
- scrolling, 229
- size
  - alternative resource qualifier, 313
  - compatibility, 301
  - configuring, 130
- touch types, 314
- workflow, 234
- website, 130

scrolling screens, 229

ScrollView control, 229

SDK (Software Development Kit)

- advantages, 25
- application framework
  - important packages, 43–45
  - third-party APIs, 45
- documentation, 43, 83–85
- download website, 52
- emulator, launching, 411
- license agreement, 42–43, 52
- Manager
  - launching, 47
  - overview, 48
- names, 25
- problems, 41
- versions, targeting, 126–128
  - maximum, 128
  - minimum, 127
  - specifying, 128
- upgrades, 29, 41

seamlessness design website, 361

searching

Eclipse, 442

preferences, 265

SearchRecentSuggestions content provider, 286, 300

security, 31

applications running as operating system users, 31

certificate signing, 32

designing, 351

developer registration, 32

permissions

ad-hoc, 31

adding, 75

application specific, 31

contacts, 292

content providers, 289

files, 277

groups, 135

INTERNET, 75

manifest file settings, 121

protection levels, 135

registering, 133-134

settings, 292

verifying, 382

voicemail, 291

website, 136

website, 136

seek bars, 194

SeekBar class, 194

<selector> tag, 156

selectors, 156

self-publishing applications, 394-395

servers

live, managing, 345

testing, 337, 370

Service class, 104, 116

services

backup, testing, 373

defined, 104

location-based, adding, 76-78

coordinates website, 413

emulator location, 77

last known location, 77

MyFirstAndroidApp, 76-78

sending coordinates, 418

overview, 116

purposes, 116

registering, 133

source control, 439

testing, 370

ServiceTestCase class, 359

setAdapter( ) method, 224

setContentView( ) method, 173

setCurrentTabByTag( ) method, 229

setFilters( ) method, 179

setIndicator( ) method, 229

setListAdapter( ) method, 225

setOnClickListener( ) method, 185

setOnItemClickListener( ) method, 224

setOnTimeChangedListener( ) method, 191

setText( ) method, 173, 178

Settings content provider, 286, 292, 300

setup( ) method, 229

ShapeDrawable class, 153

shared preferences, creating, 265

SharedPreferences class, 105, 264

SharedPreferences interface, 264
methods, 265
reference website, 273
SharedPreferences.Editor interface
methods, 266
reference website, 273
SHOP4APPS website, 393
showDialog( ) method, 254
shrinkColumns attribute, 215
signature protection level (permissions), 135
signing applications, 382-384
SimpleFragDialogActivity class, 258
size
cache partitions, 407
max VM application heap, 407
progress indicators, 192
RAM (devices), 405
screens
alternative resource qualifier, 313
compatibility, 301
configuring, 130
sliding drawers, 230
SlidingDrawer class, 230
smoke testing, 368
SMS messages, simulating, 416, 434
Snake app, 51
adding to Eclipse, 54
AVD, creating, 56-57
launch configuration, creating, 58-59
missing folder error, 56
running in emulator, 59-60
sample code website, 81
Soc.io Mall marketplace, 393
software
development kit. *See* SDK
processes, choosing. 356
iterative, 327
waterfall approaches, 326
website, 346
requirements, 37-38
upgrades, 23
source code websites, 6
source control, 339, 439
span attribute, 215
specialized testing
application popularity, increasing, 374
backup services, 373
billing, 373
conformance, 372
installations, 372
integration points, 371
internationalization, 372
performance, 373
unexpected configuration changes, 373
upgrades
applications, 371
devices, 372
specialty fragments, 237
Spinner class, 181-183
Spinner sample app, 51
SpinnerTest sample app, 51
sqlite3 tool, 99, 375
src folder, 65
src/com/androidbook/myfirstandroidapp/MyFirstAndroidAppActivity.java activity, 65
stability, 349-350
Stack Overflow website, 6

stacks (activities), 107
standalone applications, 340
start( ) method
  Chronometer control, 196
  MediaPlayer class, 75
startup options (emulator), 408
state, activity callbacks, 107-109
  initializing/retrieving activities, 109
  static data, initializing, 108
  stopping/saving/releasing, 109
static activity data, initializing, 108
statistics collection, adding, 379
stepping through code (Eclipse), 72
stop( ) method
  Chronometer control, 196
  MediaPlayer class, 75
stopping
  activity data, 109
  processes, 426
storing
  animations, 139
  application data, 275–276
  Booleans, 139
  cache files, 282–283
  capacity, 276
  colors, 139
  device data, 332
  dimensions, 139
  drawables, 139
  files, 141, 283–284
  images, 139–141
  integers, 139
  layouts, 140
  menus, 139
  mixed-type arrays, 139
  primitive resources, 140
  private data, 351
  raw files, 140
  resources, 137–140
  strings, 139
  styles, 140
  themes, 140
  XML files, 139
stretchColumns attribute, 215
StrictMode class, 373
<string> tag, 146
strings, 149
  accessing, 147–148
  bold/italic/underline, 147
  formatting, 146–147
  overview, 146
  plural, 149
  storing, 139
styles, storing, 140
subdirectories
  cache, 279
  creating, 282
  files, retrieving, 279
  retrieving, 279
summary attribute, 269
Support4Demos app, 51
Support13Demos app, 51
<supports-gl-texture> tag, 131
<supports-screen> tag, 306
Switch control, 183, 187
switches, 183, 187, 230
system
  requirements
    enforcing, 125–128
    SDK versions, targeting, 126–128
  resources, 167–168

T

T-Mobile G1, 20
tab interfaces, 226
- TabActivity class, 226-228
- TabHost class, 229

TabActivity class, 226-228, 231, 238
TabContentFactory interface, 226
TabHost class, 229-231
TableLayout class, 172, 214-216, 231
tables, 214-220
tablet compatibility, 318-319
tabs (Eclipse), closing, 441
TabSpecs class, 226
tags
- <animation-list>, 157
- <b>, 147
- <color>, 151
- <compatible-screens>, 131
- <dimen>, 152
- <drawable>, 153-154
- <fragment>, 235
- <i>, 147
- <integer>, 150
- <intent-filter>, 132
- <permission>, 135
- <PreferenceScreen>, 269
- <provider>, 133
- <selector>, 156
- <string>, 146
- <supports-gl-texture>, 131
- <supports-screens>, 306
- <TextView>, 173
- <u>, 147
- <uses-configuration>, 129
- <uses-feature>, 129-130
- <uses-library>, 130
- <uses-permission>, 134

target devices
- acquiring, 335
- identifying, 333

targeting
- markets, 344
- platforms, 328, 381
- SDK versions, 126-128
  - maximum, 128
  - minimum, 127
  - specifying, 128

tasks (Eclipse), 442
test package, 359
testing
- applications, 343
  - popularity, increasing, 374
  - usability, 370
- automating, 368
- backup services, 373
- billing, 373
- black-box, 369
- build validations, 368
- clients/servers/cloud, 337
- conformance, 372
- content providers, 285
- coverage, maximizing, 367
- defect tracking systems, 363-365
- development environment with Snake application, 53
  - adding to Eclipse, 54
  - AVD, creating, 56-57
  - launch configuration, creating, 58-59
  - running in emulator, 59-60
- devices, 336

emulator
    benefits, 368
    disadvantages, 369
environment, 365
    clean states, 366-367
    device configurations, 365-366
    real world simulations, 367
feasibility, 356
firmware upgrades, 345
hardware, obtaining, 336
installations, 372
integration points, 371
internationalization, 372
loss of signal, 367
mistakes, 375
performance, 373
personal devices, 330
plans, 338
preproduction devices, 368
real-world limitations, 337
reference websites, 376
release versions, 384
remote servers/services, 370
third-party
    documentation, 338
    firmware modifications, 365
    standards, 371
tools, 374-375
unexpected configuration changes, 373
unit, 358
upgrades
    applications, 371
    devices, 372
visual appeal, 370
white-box, 369

text
    contextual links, 174-176
    displaying, 173-174
    ellipsis (...), 174
    height, 174
    input methods alternative resource qualifier, 315
    Toast messages, 185
    width, 174
TextSwitcher class, 230
TextView class, 173
    attributes, 173-174
    contextual links, 174-176
    height, 174
    width, 174
<TextView> tag, 173
themes, 140
third-party
    APIs, 45
    content providers, 299
    device databases, 333
    firmware modifications, 365
    native applications, compared, 33
    quality standards, 352
    requirements, 330
    standards, 371
    testing documentation, 338
threads, monitoring, 426
TicTacToeLib sample app, 51
TicTacToeMain sample app, 51
time-waster games appearances, 14
TimePicker class, 191
TimePickerDialog class, 253, 260

times
- clocks
  - analog, 197
  - digital, 196
- picker dialogs, 253
- timers, 195-196
- user input, 191

title attribute, 269

title bars (progress indicators), 193

Toast messages, 185

ToggleButton control, 183, 186

toggles, 183, 186

toLeftOf attribute, 213

tools, 25, 46
- ADB, 87
- Android documentation, 83-85
- application design, 354
- AVD Manager, 47-48
- bmgr, 98, 375
- DDMS
  - debuggers, attaching, 425
  - debugging applications, 87
  - drag-and-drop operations, 433
  - Emulator Control pane, 434
  - features, 424
  - File Explorer, 430-433
  - garbage collection, 428
  - heap statistics, 427
  - HPROF files, 429
  - logging, 436-437
  - memory allocations, 430
  - perspective (Eclipse), 47
  - processes, stopping, 426
  - screen captures, 435
  - standalone application, 423
  - threads, monitoring, 426
  - website, 100
- development, leveraging, 360
- dmtracedump, 98
- Eclipse ADT plug-in, 46-47, 88
  - resources, 89, 143-146, 164-166
  - UI designer, 90
- emulator, 49
  - AVDs, 401-407
  - benefits, 368
  - best practices, 399-401
  - calling between, 413
  - console, 416-419
  - Control pane, 434
  - disadvantages, 369
  - files, managing, 432-433
  - GPS location, 411-413
  - Hierarchy Viewer, launching, 92
  - icon tips, 419
  - launching, 60, 407-411
  - limitations, 420-421
  - location, configuring, 77
  - messaging between, 415
  - MyFirstAndroidApp, 67-73
  - overview, 85, 399
  - pages, adding, 419
  - PeakBagger 1.0, 86
  - performance, 407-408
  - screen captures, 435
  - smartphone-style example, 49
  - Snake app, running, 59-60
  - startup options, 408
  - tablet-style example, 50
  - tips, 419
  - unlocking, 60

wallpaper, editing, 419
website, 86, 100, 421
etc1tool, 99
Exerciser Monkey, 373, 376
Extract Local Variable, 445
Extract Method, 445
Hierarchy Viewer
launching, 92
layout controls above application content, 92
Layout View mode, 92-93
modes, 92
online tutorial, 94
overview, 91
Pixel Perfect mode, 94
user interface optimization, 94
hprof-conv, 98
layoutopt, 220, 375
logcat, 99, 374, 436-437
mksdcard, 99
monkey, 99
monkeyrunner, 99
Nine-Patch Stretchable Graphics, 95-97
Rename, 444
resource editors, 89
sample applications, 50-51
SDK Manager, 47-48
signing, 384
sqlite3, 99, 375
testing, 374-375
traceview, 98, 375
UI designer, 90
website, 98
zipalign, 99

toRightOf attribute, 213
touch screen support, 405, 314
TouchUtils class, 359
traceview tool, 98, 375
trackball support, 406
tracking
devices, 332
memory allocations, 430
transitions (intents), 104
action/data, 114
additional information, adding, 115
application navigation, 115-116
broadcasting, 117
filters, 114, 132-133
Google common, 115
launching activities
by class name, 113
external, 114
primary entry points, choosing, 132
website, 118
troubleshooting. *See also* debugging
alternative resource performance issues, 317
build errors, 447
crashes, 345
layouts, 220
SDK problems, 41
single device bugs, 359
Snake application missing folder, 56
user interfaces
drawing issues, 92-93
layout control organization, 94
trust relationships, 32
tweened animations, 156
accessing, 158
defining, 157

overview, 157
spinning graphic example, 158

types
animations, 156
buttons, 183
defects, 364-365
dialogs, 252-253
fragments, 237
layouts, 205
resources, 138-140

U

<u> tag, 147
UI designer, 88-90
underlining strings, 147
unexpected configuration changes, testing, 373
Uniform Resource Identifiers (URIs), 31
unit testing, 358
unlocking, emulator, 60
unregisterOnSharedPreferenceChange-Listener( ) method, 267
updating
contacts, 298
designing for, 354
fragments, 236
preferences, 266-267

upgrades, 23
applications, 371, 392
designing for, 354
devices, 372
firmware, testing, 345
SDKs, 41, 29

uploading (Android Market)
applications, 387
marketing assets, 388

Uri class
methods, 74
website, 81

URIs (Uniform Resource Identifiers), 31
usability testing, 370
use cases, creating, 329
user-facing features, 23
user interfaces
autocompletion, 179-181
buttons, 183
basic, 183-184
check boxes, 183, 186
image buttons, 185
radio buttons, 183, 187-189
switches, 183, 187
toggles, 183, 186
compatibility, 303-304
best practices, 322
fragments, 305
nine-patch stretchable graphics, 306
screen types, 305-306
Support (Compatibility) Package, 305
working squares, 306-307
containers, 221
controls
App Widgets, compared, 172
defined, 171
layout, 172
data-driven controls, 221-222
adapters, 222
arrays, 222-223
click events, handling, 224
data, binding, 224
databases, 223-224
headers/footers, 226
lists of items, 225

date input, 190–191
debugging and profiling website, 100
designing, 348–349
dialogs
  adding, 251–254
  alerts, 252
  basic, 252
  character pickers, 252
  customizing, 256
  date pickers, 253
  defining, 254
  destroying, 255–256
  dismissing, 255
  fragment method, 257–260
  initializing, 254
  launching, 255
  lifecycle, 254
  progress, 253
  time pickers, 253
  types, 252–253
  websites, 260
documentation, 338
drawing issues, debugging, 92–93
fragments
  Activity classes, creating, 246–247
  applications, designing, 238
  attaching, 237
  compatibility, 305
  creating, 237
  defined, 104, 111
  defining, 235
  destroying, 237
  detaching, 237
  layout resource files, creating, 244–246
  legacy support, 247–248
  lifecycle, 235
  ListFragment, implementing, 240–243
  lists, 238
  MP3 music player example, 111–113
  overview, 233
  pausing, 237
  resuming activity, 237
  screen workflow, 234
  specialty, 237
  stopping activity, 237
  updates, 236
  user preferences, 238
  visibility, 237
  web content, 238
  website, 118
  WebView, implementing, 243–244
  workflow flexibility, improving, 111–113
galleries, 205, 222
grids, 222
guidelines website, 361
indicators
  adjusting progress, 194
  clocks, 196–197
  progress bars, 192–193
  ratings, 194–195
  time passage, 195–196
inspecting at pixel level, 94
layouts
  attributes, 205–206
  creating programmatically, 201–203
  creating XML, 199–200

declaring website, 231
frames, 207-209
grids, 216-220
inspecting, 220
linear, 206, 209-210
multiple, 220
nesting, 207
organization, 94
relative, 211-214
tables, 214-216
types, 205
lists, 222
Nine-Patch Stretchable Graphics, 95
accessing, 155
creating, 95-97, 155
defined, 95, 155
device compatibility, 306
scaling, 95
website, 100
optimizing, 94
scrolling, 229
sliding drawers, 230
spinners, 181-183
switchers, 230
tabs, 226
TabActivity class, 226-228
TabHost control, 229
text, displaying, 173
contextual links, 174-176
ellipsis, 174
height, 174
width, 174
time input, 191
user input
autocompletion, 179-181
dates, 190-191
filters, 178-179
hints, 177
input box height, 177
retrieving, 176-178
spinners, 181-183
times, 191
View class, 171, 204
ViewGroups, 204

UserDictionary content provider, 286, 291, 300

users
demands, meeting, 348
input, 176
autocompletion, 179-181
dates, 190-191
filters, 178-179
hints, 177
input box height, 177
retrieving, 176-178
spinners, 181-183
times, 191
preferences
fragments, 238
loading up, 270-272
resource files, creating, 269-270

<uses-configuration> tag, 129

<uses-feature> tag, 129-130

<uses-library> tag, 130

<uses-permission> tag, 134

V

V CAST Apps website, 393

versioning

- applications, 125
- code, 380
- schemes, choosing, 339-340

Videos section (documentation), 84

View class, 171, 204

- adding to ViewGroup, 204
- containers, 204
- galleries, 205
- grouping, 204

View containers

- sliding drawers, 230
- switchers, 230
- tabs, 228

view package, 171

ViewAsserts class, 359

ViewGroup class

- layout attributes, 205-206
- layout subclasses, 204
- view containers, 204
- View class, compared, 204
- website, 231

views

- AdapterView controls, 222
- containers, 221
- data-driven containers, 221-222
  - adapters, 222
  - arrays, 222-223
  - click events, handling, 224
  - data, binding, 224
  - databases, 223-224
  - lists of items, 225
- FrameLayout, 207-209
- GridLayout, 216-220
- LinearLayout, 209-210
- RelativeLayout, 211-214
- scrolling support, 229
- TableLayout, 214-216
- tabs, 226-227

ViewSwitcher class, 230

virtual device management website, 421

visual appeal, testing, 370

VMs (virtual machines), 31

voicemail, accessing, 291

VoicemailContract content provider, 286, 291

W

wallpaper, editing, 419

WAP (Wireless Application Protocol), 15

- browsers, 16
- commercializing, 16
- introduction, 15

waterfall development, 326, 346

web

- application development, 33
- content, 238

websites

- Activity class, 81, 118
- ad revenue, 379
- ADB, 88, 100
- Adobe AIR, 33
- AlarmClock content provider, 300
- AlertDialog class, 260
- alternative marketplaces, 394
- alternative resources, 321
- Amazon Appstore, 393
- anddev.org forum, 7

Android
  Developers, 6, 350
  Development, 35
  documentation, 83
  Tools Project Site, 7
API levels, 136
book, 6
Browser content provider, 300
CalendarContract content provider, 291
CallLog content provider, 300
CharacterPickerDialog class, 260
CheckBoxPreference class, 274
"Common Tasks and How to Do Them in Android," 81
compatibility
  best practices, 322
  package, 118
<compatible-screens> tag, 131
Conder blog, 8
Contacts content provider, 300
ContactsContract content provider, 297, 300
Context class, 118, 284
country codes, 322
Darcey blog, 8
DatePickerDialog class, 260
DDMS, 100
design best practices, 361
Developer.com, 7
"Developing on a Device," 81
development
  system requirements, 38
  tools, 98
Dialog class, 260, 321
DialogFragment class, 250, 260
dialogs, 260
Eclipse, 52
EditTextPreference class, 274
emulator, 86, 100, 421
Environment class, 284
Exerciser Monkey, 376
extreme programming, 346
FierceDeveloper, 7
File class, 284
Fragment class, 118, 250
fragments, 118
FrameLayout class, 231
Gallery class, 231
Google
  Analytics on Android, 361
  Android Developer's Guide, 52, 100
  common intents, 115
  Experience devices, 335
  Maps, 413
  Team Android Apps, 7
  TV media formats, 321
GridLayout class, 231
GridView class, 231
Handango, 393
Hierarchy Viewer online tutorial, 94
IDEs besides Eclipse, 38
in-application billing, 379
installation instructions, 39
intents, 118
Issue Tracker, 41
iterative development, 346
Java Platform, Standard Edition, 52

java.io package, 284
language codes, 322
layouts, declaring, 231
LinearLayout class, 231
Linux configuration, 39
ListActivity class, 231
ListFragment class, 250
ListPreference class, 274
ListView class, 231
LocationManager, 81
Log class, 81
LVL, 378
manifest file reference, 136
Market, 7, 385, 395
  content guidelines, 390
  help, 390
  fees, 390
  filters, 381, 395
  reviews, 344
MediaPlayer class, 81
MediaStore content provider, 300
memory allocations, tracking, 430
Mobiletuts+, 7
multiple screens, 136, 322
Nine-Patch Stretchable Graphics, 100
Official Android Developers blog, 35
OHA, 6, 35
<permission> tag, 135
permissions, 136
pre-publication checklist, 361
Preference class, 274
PreferenceActivity class, 273
PreferenceCategory class, 274
PreferenceFragment class, 250
PreferenceScreen class, 274
ProgressDialog class, 260
provider package, 300
Rapid Application Development (RAD), 346
RelativeLayout class, 231
resources, 168
responsiveness design, 361
rs:ResEnum app, 168
runtime changes, handling, 321
screens
  compatibility mode, 306, 321
  support, 130
SDK
  download, 52
  License Agreement, 52
  upgrades, 41
seamlessness design, 361
SearchRecentSuggestions content provider, 300
security, 136
Settings content provider, 300
SharedPreferences interface, 273
SharedPreferences.Editor interface, 273
SHOP4APPS, 393
signing applications, 384
Snake application sample code, 81
Soc.io Mall, 393
software development process, 346
source code, 6
Stack Overflow, 6
storage, 284
StrictMode class, 373
Support4Demos sample app, 51
Support13Demos sample app, 51
<supports-gl-texture> tag, 131

TabActivity class, 231
TabHost class, 231
TableLayout class, 231
testing references, 376
third-party content providers, 299
TimePickerDialog class, 260
Uri class, 81
user interfaces
debugging and profiling, 100
guidelines, 361
layout controls, 94
UserDictionary content provider, 300
<uses-configuration> tag, 129
<uses-feature> tag, 130
V CAST Apps, 393
ViewGroup class, 231
virtual device management, 421
VoicemailContract class, 292
waterfall development, 346
WebViewFragment class, 250
Windows USB driver, 39
Wireless Developer Network, 7
WURFL, 333
XDA-Developers Android forum, 7

WebView classes, implementing, 243-244

WebViewFragment class, 238, 250

weight attribute, 210

weightSum attribute, 210

white-box testing, 369

widget package, 171

windows (Eclipse)
maximizing, 440
minimizing, 440
open. limiting, 441
side by side view, 440

Windows USB driver download website, 39

Wireless Application Protocol (WAP), 15-16

Wireless Developer Network website, 7

workflow
flexibility, improving, 111-113
screens, 234

working squares, 306-307

workspace (Eclipse)
perspectives, 440
source control services, 439
tabs, closing, 441
two sections of same file, viewing, 441
windows
limiting open, 441
maximizing/minimizing, 440
side by side view, 440

writeable files, 277

writing files, 279

WURFL website, 333

X–Z

XDA-Developers Android forum, 7

XML
escaping, 147
files, 159
accessing, 160
defining, 160
parsing utilities, 280
reading, 280-281
storing, 139
layout resource files, creating, 199-200
pull parsing package, 45
SAX support package, 45

zipalign tool, 99