Android™ Wireless Application Development

Second Edition
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
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data:
Conder, Shane, 1975-
Android wireless application development / Shane Conder, Lauren Darcey. — 1st ed.
p. cm.
QA76.76.A65C6637 2011
005.1—dc22
2010046618

Copyright © 2011 Shane Conder and Lauren Darcey

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. For information regarding permissions, write to:

Pearson Education, Inc
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

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/).


Text printed in the United States on recycled paper at Edwards Brothers, Ann Arbor, Michigan

Second Printing: March 2011
This book is dedicated to Bit, Nibble, Stack, Queue, Heap, and Null.
Contents at a Glance

Introduction  1

I: An Overview of Android

1 Introducing Android  7
2 Setting Up Your Android Development Environment  29
3 Writing Your First Android Application  43

II: Android Application Design Essentials

4 Understanding the Anatomy of an Android Application  69
5 Defining Your Application Using the Android Manifest File  81
6 Managing Application Resources  97

III: Android User Interface Design Essentials

7 Exploring User Interface Screen Elements  133
8 Designing User Interfaces with Layouts  173
9 Drawing and Working with Animation  205

IV: Using Common Android APIs

10 Using Android Data and Storage APIs  231
11 Sharing Data Between Applications with Content Providers  259
12 Using Android Networking APIs  287
13 Using Android Web APIs  301
14 Using Location-Based Services (LBS) APIs  315
15 Using Android Multimedia APIs  335
16 Using Android Telephony APIs  353
17 Using Android 3D Graphics with OpenGL ES 367
18 Using the Android NDK 397
19 Using Android’s Optional Hardware APIs 407

V: More Android Application Design Principles
20 Working with Notifications 423
21 Working with Services 437
22 Extending Android Application Reach 451
23 Managing User Accounts and Synchronizing User Data 489
24 Handling Advanced User Input 499
25 Targeting Different Device Configurations and Languages 523

VI: Deploying Your Android Application to the World
26 The Mobile Software Development Process 551
27 Designing and Developing Bulletproof Android Applications 571
28 Testing Android Applications 585
29 Selling Your Android Application 597

VII: Appendixes
A The Android Emulator Quick-Start Guide 613
B The Android DDMS Quick-Start Guide 635
C The Android Debug Bridge Quick-Start Guide 647
D Eclipse IDE Tips and Tricks 661
E The SQLite Quick-Start Guide 669
Index 683
Table of Contents

Introduction  1
   Who Should Read This Book  1
   Key Questions Answered in This Book  2
   How This Book Is Structured  2
   An Overview of Changes in This Edition  3
   Development Environment Used in This Book  4
   Supplementary Materials Available  5
   Where to Find More Information  5
   Conventions Used in This Book  6
   Contacting the Authors  6

I: An Overview of Android

1 Introducing Android  7
   A Brief History of Mobile Software Development  7
      Way Back When  7
      “The Brick”  9
      Wireless Application Protocol (WAP)  11
      Proprietary Mobile Platforms  13
   The Open Handset Alliance  15
      Google Goes Wireless  15
      Forming the Open Handset Alliance  15
      Manufacturers: Designing the Android Handsets  16
      Mobile Operators: Delivering the Android Experience  17
      Content Providers: Developing Android Applications  17
      Taking Advantage of All Android Has to Offer  18
   Android Platform Differences  18
      Android: A Next-Generation Platform  18
      Free and Open Source  20
      Familiar and Inexpensive Development Tools  20
      Reasonable Learning Curve for Developers  20
      Enabling Development of Powerful Applications  21
      Rich, Secure Application Integration  21
      No Costly Obstacles to Publication  21
2 Setting Up Your Android Development Environment 29

Configuring Your Development Environment 29
   Configuring Your Operating System for Device Debugging 30
   Configuring Your Android Hardware for Debugging 30
Upgrading the Android SDK 31
Problems with the Android Software Development Kit 32
Exploring the Android SDK 32
   Understanding the Android SDK License Agreement 32
   Reading the Android SDK Documentation 33
Exploring the Android Application Framework 35
Getting to Know the Android Tools 35
Exploring the Android Sample Applications 40
Summary 41
References and More Information 41

3 Writing Your First Android Application 43

Testing Your Development Environment 43
   Adding the Snake Application to a Project in Your Eclipse Workspace 43
Creating an Android Virtual Device (AVD) for Your Snake Project 44
Creating a Launch Configuration for Your Snake Project 46
Running the Snake Application in the Android Emulator 47
Building Your First Android Application  48
  Creating and Configuring a New Android Project  50
  Core Files and Directories of the Android Application  50
  Creating an AVD for Your Project  51
  Creating Launch Configurations for Your Project  52
  Running Your Android Application in the Emulator  53
  Debugging Your Android Application in the Emulator  56
  Adding Logging Support to Your Android Application  59
  Adding Some Media Support to Your Application  60
  Adding Location-Based Services to Your Application  62
  Debugging Your Application on the Hardware  65
Summary  66
References and More Information  67

II: Android Application Design Essentials

4 Understanding the Anatomy of an Android Application  69
  Mastering Important Android Terminology  69
  Using the Application Context  70
    Retrieving the Application Context  70
    Using the Application Context  70
  Performing Application Tasks with Activities  71
    The Lifecycle of an Android Activity  72
    Managing Activity Transitions with Intents  76
  Working with Services  78
  Receiving and Broadcasting Intents  79
Summary  80
References and More Information  80

5 Defining Your Application Using the Android Manifest File  81
  Configuring the Android Manifest File  81
    Editing the Android Manifest File  82
Handling User Events 161
  Listening for Touch Mode Changes 161
  Listening for Events on the Entire Screen 162
  Listening for Long Clicks 163
  Listening for Focus Changes 164
Working with Dialogs 165
  Exploring the Different Types of Dialogs 165
  Tracing the Lifecycle of a Dialog 166
  Working with Custom Dialogs 168
Working with Styles 168
Working with Themes 170
Summary 171

8 Designing User Interfaces with Layouts 173
Creating User Interfaces in Android 173
  Creating Layouts Using XML Resources 173
  Creating Layouts Programmatically 175
Organizing Your User Interface 177
  Understanding View versus ViewGroup 178
Using Built-In Layout Classes 181
  Using FrameLayout 183
  Using LinearLayout 185
  Using RelativeLayout 186
  Using TableLayout 190
  Using Multiple Layouts on a Screen 192
Using Built-In View Container Classes 192
  Using Data-Driven Containers 194
  Organizing Screens with Tabs 198
  Adding Scrolling Support 201
  Exploring Other View Containers 202
Summary 203

9 Drawing and Working with Animation 205
Drawing on the Screen 205
  Working with Canvases and Paints 205
Working with Text 210
  Using Default Fonts and Typefaces 210
  Using Custom Typefaces 211
  Measuring Text Screen Requirements 212
Working with Bitmaps 212
  Drawing Bitmap Graphics on a Canvas 213
  Scaling Bitmap Graphics 213
  Transforming Bitmaps Using Matrixes 213
Working with Shapes 214
  Defining Shape Drawables as XML Resources 214
  Defining Shape Drawables Programmatically 215
  Drawing Different Shapes 215
Working with Animation 221
  Working with Frame-by-Frame Animation 223
  Working with Tweened Animations 224
Summary 230

IV: Using Common Android APIs

10 Using Android Data and Storage APIs 231
  Working with Application Preferences 231
    Creating Private and Shared Preferences 232
    Searching and Reading Preferences 232
    Adding, Updating, and Deleting Preferences 233
    Finding Preferences Data on the Android File System 234
  Working with Files and Directories 235
    Exploring with the Android Application Directories 235
    Working with Other Directories and Files on the Android File System 238
  Storing Structured Data Using SQLite Databases 239
    Creating a SQLite Database 240
    Creating, Updating, and Deleting Database Records 242
    Querying SQLite Databases 244
    Closing and Deleting a SQLite Database 250
    Designing Persistent Databases 250
    Binding Data to the Application User Interface 253
Summary 257
References and More Information 258
11 Sharing Data Between Applications with Content Providers 259
Exploring Android’s Content Providers 259
   Using the MediaStore Content Provider 260
   Using the CallLog Content Provider 261
   Using the Browser Content Provider 263
   Using the Contacts Content Provider 264
   Using the UserDictionary Content Provider 267
   Using the Settings Content Provider 267
Modifying Content Providers Data 267
   Adding Records 267
   Updating Records 268
   Deleting Records 269
Enhancing Applications Using Content Providers 269
   Accessing Images on the Device 270
Acting as a Content Provider 274
   Implementing a Content Provider Interface 275
   Defining the Data URI 276
   Defining Data Columns 276
   Implementing Important Content Provider Methods 276
   Updating the Manifest File 282
Working with Live Folders 282
Summary 285
References and More Information 285

12 Using Android Networking APIs 287
Understanding Mobile Networking Fundamentals 287
Accessing the Internet (HTTP) 288
   Reading Data from the Web 288
   Using HttpURLConnection 289
Parsing XML from the Network 290
Processing Asynchronously 291
Working with AsyncTask 292
Using Threads for Network Calls 293
Displaying Images from a Network Resource 295
Retrieving Android Network Status 297
Contents

Summary 298
References and More Information 299

13 Using Android Web APIs 301
Browsing the Web with WebView 301
  Designing a Layout with a WebView Control 302
  Loading Content into a WebView Control 302
  Adding Features to the WebView Control 304
Building Web Extensions Using WebKit 307
  Browsing the WebKit APIs 307
  Extending Web Application Functionality to Android 308
Working with Flash 311
  Enabling Flash Applications 312
  Building AIR Applications for Android 313
Summary 314
References and More Information 314

14 Using Location-Based Services (LBS) APIs 315
Using Global Positioning Services (GPS) 315
  Using GPS Features in Your Applications 316
  Finding Your Location 316
  Locating Your Emulator 318
Geocoding Locations 318
Mapping Locations 322
  Mapping Intents 322
  Mapping Views 322
Getting Your Debug API Key 325
Panning the Map View 326
Zooming the Map View 327
Marking the Spot 327
Doing More with Location-Based Services 332
Summary 333
References and More Information 333
15 Using Android Multimedia APIs 335
  Working with Multimedia 335
  Working with Still Images 336
    Capturing Still Images Using the Camera 336
    Configuring Camera Mode Settings 340
    Sharing Images 341
    Assigning Images as Wallpapers 342
  Working with Video 343
    Recording Video 343
    Playing Video 345
  Working with Audio 346
    Recording Audio 347
    Playing Audio 348
    Sharing Audio 349
    Searching for Multimedia 350
  Working with Ringtones 351
Summary 351
References and More Information 351

16 Using Android Telephony APIs 353
  Working with Telephony Utilities 353
    Gaining Permission to Access Phone State Information 354
    Requesting Call State 354
    Requesting Service Information 356
    Monitoring Signal Strength and Data Connection Speed 356
  Working with Phone Numbers 357
Using SMS 357
    Gaining Permission to Send and Receive SMS Messages 358
    Sending an SMS 358
    Receiving an SMS 360
  Making and Receiving Phone Calls 362
    Making Phone Calls 362
    Receiving Phone Calls 364
Summary 365
References and More Information 365
17 Using Android 3D Graphics with OpenGL ES 367
Working with OpenGL ES 367
  Leveraging OpenGL ES in Android 368
  Ensuring Device Compatibility 368
Using OpenGL ES APIs in the Android SDK 369
Handling OpenGL ES Tasks Manually 369
  Creating a SurfaceView 370
  Starting Your OpenGL ES Thread 371
  Initializing EGL 373
  Initializing GL 374
  Drawing on the Screen 375
Drawing 3D Objects 376
  Drawing Your Vertices 376
  Coloring Your Vertices 377
  Drawing More Complex Objects 378
  Lighting Your Scene 379
  Texturing Your Objects 381
Interacting with Android Views and Events 383
  Enabling the OpenGL Thread to Talk to the Application Thread 384
  Enabling the Application Thread to Talk to the OpenGL Thread 386
Cleaning Up OpenGL ES 387
Using GLSurfaceView (Easy OpenGL ES) 388
Using OpenGL ES 2.0 391
  Configuring Your Application for OpenGL ES 2.0 391
  Requesting an OpenGL ES 2.0 Surface 391
Summary 395
References and More Information 396

18 Using the Android NDK 397
Determining When to Use the Android NDK 397
Installing the Android NDK 398
Exploring the Android NDK 398
  Running an Android NDK Sample Application 399
Creating Your Own NDK Project 399
  Calling Native Code from Java 400
  Handling Parameters and Return Values 401
  Using Exceptions with Native Code 402
19  Using Android’s Optional Hardware APIs  

Interacting with Device Hardware  

Using the Device Sensor  

   Working with Different Sensors  
   Acquiring Access to a Sensor  
   Reading Sensor Data  
   Calibrating Sensors  
   Determining Device Orientation  
   Finding True North  

Working with Wi-Fi  

Working with Bluetooth  

   Checking for the Existence of Bluetooth Hardware  
   Enabling Bluetooth  
   Querying for Paired Devices  
   Discovering Devices  
   Establishing Connections Between Devices  

Monitoring the Battery  

Summary  

References and More Information  

V: More Android Application Design Principles  

20  Working with Notifications  

Notifying the User  

Notifying with the Status Bar  

   Using the NotificationManager Service  
   Creating a Simple Text Notification with an Icon  
   Working with the Notification Queue  
   Updating Notifications  
   Clearing Notifications  

Vibrating the Phone  

Blinking the Lights  

Making Noise
## 21 Working with Services 437

- Determining When to Use Services 437
- Understanding the Service Lifecycle 438
- Creating a Service 438
- Controlling a Service 443
- Implementing a Remote Interface 444
- Implementing a Parcelable Class 446

## 22 Extending Android Application Reach 451

- Enhancing Your Applications 451
- Working with App Widgets 452
  - Creating an App Widget 453
  - Installing an App Widget 460
  - Becoming an App Widget Host 460
- Working with Live Wallpapers 461
  - Creating a Live Wallpaper 462
  - Installing a Live Wallpaper 465
- Acting as a Content Type Handler 466
- Determining Intent Actions and MIME Types 467
  - Implementing the Activity to Process the Intents 468
  - Registering the Intent Filter 469
- Making Application Content Searchable 469
  - Enabling Searches Within Your Application 470
  - Enabling Global Search 478
- Working with Live Folders 480
  - Creating Live Folders 481
  - Installing a Live Folder 485

## Contents

- Customizing the Notification 432
- Designing Useful Notifications 434
- Summary 434
- References and More Information 435

- 21 Working with Services 437
  - Determining When to Use Services 437
  - Understanding the Service Lifecycle 438
  - Creating a Service 438
  - Controlling a Service 443
  - Implementing a Remote Interface 444
  - Implementing a Parcelable Class 446
  - Summary 449
  - References and More Information 449

- 22 Extending Android Application Reach 451
  - Enhancing Your Applications 451
  - Working with App Widgets 452
    - Creating an App Widget 453
    - Installing an App Widget 460
    - Becoming an App Widget Host 460
  - Working with Live Wallpapers 461
    - Creating a Live Wallpaper 462
    - Installing a Live Wallpaper 465
  - Acting as a Content Type Handler 466
  - Determining Intent Actions and MIME Types 467
    - Implementing the Activity to Process the Intents 468
    - Registering the Intent Filter 469
  - Making Application Content Searchable 469
    - Enabling Searches Within Your Application 470
    - Enabling Global Search 478
  - Working with Live Folders 480
    - Creating Live Folders 481
    - Installing a Live Folder 485
  - Summary 487
  - References and More Information 487
23 Managing User Accounts and Synchronizing User Data 489
Managing Accounts with the Account Manager 489
Synchronizing Data with Sync Adapters 490
Using Backup Services 491
Choosing a Remote Backup Service 492
Implementing a Backup Agent 492
Backing Up and Restoring Application Data 496
Summary 497
References and More Information 497

24 Handling Advanced User Input 499
Working with Textual Input Methods 499
Working with Software Keyboards 499
Working with Text Prediction and User Dictionaries 502
Exploring the Accessibility Framework 502
Leveraging Speech Recognition Services 503
Leveraging Text-To-Speech Services 506
Working with Gestures 508
Detecting User Motions Within a View 509
Handling Common Single-Touch Gestures 509
Handling Common Multi-Touch Gestures 516
Making Gestures Look Natural 518
Working with the Trackball 519
Handling Screen Orientation Changes 519
Summary 522
References and More Information 522

25 Targeting Different Device Configurations and Languages 523
Maximizing Application Compatibility 523
Designing User Interfaces for Compatibility 525
Supporting Specific Screen Types 526
Working with Nine-Patch Stretchable Graphics 526
Using the Working Square Principle 528
Providing Alternative Application Resources 531
Working with Alternative Resource Qualifiers 531
Providing Resources for Different Orientations 537
VI: Deploying Your Android Application to the World

26 The Mobile Software Development Process 551

An Overview of the Mobile Development Process 551

Choosing a Software Methodology 552

Understanding the Dangers of Waterfall Approaches 552

Understanding the Value of Iteration 553

Gathering Application Requirements 553

Determining Project Requirements 553

Developing Use Cases for Mobile Applications 555

Incorporating Third-Party Requirements 555

Managing a Device Database 555

Assessing Project Risks 558

Identifying Target Devices 558

Acquiring Target Devices 560

Determining Feasibility of Application Requirements 561

Understanding Quality Assurance Risks 561

Writing Essential Project Documentation 562

Developing Test Plans for Quality Assurance Purposes 562

Providing Documentation Required by Third Parties 563

Providing Documentation for Maintenance and Porting 563

Leveraging Configuration Management Systems 563

Choosing a Source Control System 563
Implementing an Application Version System That Works 564
Designing Mobile Applications 564
   Understanding Mobile Device Limitations 564
   Exploring Common Mobile Application Architectures 564
   Designing for Extensibility and Maintenance 565
   Designing for Application Interoperability 566
Developing Mobile Applications 567
Testing Mobile Applications 567
Deploying Mobile Applications 568
   Determining Target Markets 568
Supporting and Maintaining Mobile Applications 568
   Track and Address Crashes Reported by Users 569
   Testing Firmware Upgrades 569
   Maintaining Adequate Application Documentation 569
   Managing Live Server Changes 569
   Identifying Low-Risk Porting Opportunities 569
Summary 570
References and More Information 570

27 Designing and Developing Bulletproof Android Applications 571
Best Practices in Designing Bulletproof Mobile Applications 571
   Meeting Mobile Users' Demands 572
   Designing User Interfaces for Mobile Devices 572
   Designing Stable and Responsive Mobile Applications 573
   Designing Secure Mobile Applications 574
   Designing Mobile Applications for Maximum Profit 575
   Leveraging Third-Party Standards for Android Application Design 576
   Designing Mobile Applications for Ease of Maintenance and Upgrades 576
   Leveraging Android Tools for Application Design 578
Avoiding Silly Mistakes in Android Application Design 578
Best Practices in Developing Bulletproof Mobile Applications 579
Designing a Development Process That Works for Mobile Development 579
Testing the Feasibility of Your Application Early and Often 579
Using Coding Standards, Reviews, and Unit Tests to Improve Code Quality 580
Handling Defects Occurring on a Single Device 582
Leveraging Android Tools for Development 583
Avoiding Silly Mistakes in Android Application Development 583
Summary 583
References and More Information 584

28 Testing Android Applications 585
Best Practices in Testing Mobile Applications 585
Designing a Mobile Application Defect Tracking System 585
Managing the Testing Environment 587
Maximizing Testing Coverage 589
Leveraging Android Tools for Android Application Testing 595
Avoiding Silly Mistakes in Android Application Testing 595
Outsourcing Testing Responsibilities 596
Summary 596
References and More Information 596

29 Selling Your Android Application 597
Choosing the Right Distribution Model 597
Packaging Your Application for Publication 598
Preparing Your Code to Package 599
Packing and Signing Your Application 600
Testing the Release Version of Your Application Package 603
Certifying Your Android Application 603
Distributing Your Applications 603
Selling Your Application on the Android Market 603
Selling Your Application on Your Own Server 609
Application 638
Prompting Garbage Collection (GC) 639
Monitoring Heap Activity 639
Monitoring Memory Allocation 640
Stopping a Process 640
Working with the File Explorer 641
Browsing the File System of an Emulator or Device 641
Copying Files from the Emulator or Device 641
Copying Files to the Emulator or Device 642
Deleting Files on the Emulator or Device 642
Working with the Emulator Control 642
Simulating Incoming Voice Calls 643
Simulating Incoming SMS Messages 643
Sending a Location Fix 643
Working with Application Logging 644
Taking Screen Captures of Emulator and Device Screens 645

C The Android Debug Bridge Quick-Start Guide 647
Listing Connected Devices and Emulators 647
Directing ADB Commands to Specific Devices 648
Starting and Stopping the ADB Server 648
Stopping the ADB Server Process 648
Starting and Checking the ADB Server Process 648
Issuing Shell Commands 649
Issuing a Single Shell Command 649
Using a Shell Session 649
Using the Shell to Start and Stop the Emulator 649
Copying Files 650
Sending Files to a Device or Emulator 650
Retrieving Files from a Device or Emulator 650
Installing and Uninstalling Applications 651
Installing Applications 651
Reinstalling Applications 651
Uninstalling Applications 651
Working with LogCat Logging 652
Displaying All Log Information 652
Including Date and Time with Log Data 652
Filtering Log Information 652
Clearing the Log 654
Redirecting Log Output to a File 654
Accessing the Secondary Logs 654
Controlling the Backup Service 654
Forcing Backup Operations 655
Forcing Restore Operations 655
Wiping Archived Data 655
Generating Bug Reports 655
Using the Shell to Inspect SQLite Databases 656
Using the Shell to Stress Test Applications 656
Letting the Monkey Loose on Your Application 656
Listening to Your Monkey 656
Directing Your Monkey’s Actions 657
Training Your Monkey to Repeat His Tricks 658
Keeping the Monkey on a Leash 658
Learning More About Your Monkey 659
Installing Custom Binaries via the Shell 659
Exploring Other ADB Commands 660

D Eclipse IDE Tips and Tricks 661
Organizing Your Eclipse Workspace 661
Integrating with Source Control Services 661
Repositioning Tabs Within Perspectives 661
Maximizing Windows 662
Minimizing Windows 662
Viewing Windows Side by Side 662
Viewing Two Sections of the Same File 662
Closing Unwanted Tabs 662
Keeping Windows Under Control 663
Creating Custom Log Filters 663
Writing Code in Java 663
Using Auto-Complete 664
Formatting Code 664
Creating New Classes 664
Creating New Methods 664
E The SQLite Quick-Start Guide 669
Exploring Common Tasks with SQLite 669
Using the sqlite3 Command-Line Interface 670
  Launching the ADB Shell 670
  Connecting to a SQLite Database 670
  Exploring Your Database 671
  Importing and Exporting the Database and Its Data 672
  Executing SQL Commands on the Command Line 674
  Using Other sqlite3 Commands 675
  Understanding SQLite Limitations 675
Learning by Example: A Student Grade Database 675
  Designing the Student Grade Database Schema 676
  Creating Simple Tables with AUTOINCREMENT 676
  Inserting Data into Tables 677
  Querying Tables for Results with SELECT 677
  Using Foreign Keys and Composite Primary Keys 678
  Altering and Updating Data in Tables 679
  Querying Multiple Tables Using JOIN 680
  Using Calculated Columns 680
  Using Subqueries for Calculated Columns 682
  Deleting Tables 682
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 and Charles Stearns (as well as Dan Galpin, Tony Hillerson, and Ronan Schwarz, who reviewed the first edition). 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, iPhone, 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, iPhone, 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 phone, laptop, 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—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.
Pioneered by the Open Handset Alliance and Google, Android is a hot, young, free, open source mobile platform making waves in the wireless world. This book provides 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, this book provides everything you need to make a smooth transition from traditional software development to mobile development—specifically, its most promising new platform: Android.

Who Should Read This Book

This book includes tips for successful mobile development based on our years in the mobile industry and covers everything you need 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 can work 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 primarily targeted at software developers with Java experience but not necessarily 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 a software developer, but also for the corporation looking at potential vertical market applications, the entrepreneur thinking about a cool phone application, and hobbyists looking for some fun with their 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 this book to use for fun and profit.

Key Questions Answered in This Book

This 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 the second edition of the book?

How This Book Is Structured

This book is divided into seven parts. The first five parts are primarily of interest to developers; Parts VI and VII provide lots of helpful information for project managers and quality assurance personnel as well as developers.
Here is an overview of the various parts in this book:

- **Part I: An Overview of Android**
  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 Design Essentials**
  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 basic drawing and animation abilities provided in the Android SDK.

- **Part IV: Using Common Android APIs**
  Part IV is a series of chapters, each devoted to a deeper understanding of the most important APIs within the Android SDK, such as the data and storage APIs (including file and database usage as well as content providers), networking, telephony, Location-Based Services (LBS), multimedia and 3D graphics APIs, and the optional hardware APIs available.

- **Part V: More Android Application Design Principles**
  Part V covers more advanced Android application design principles, such as notifications and services.

- **Part VI: Deploying Your Android Application to the World**
  Part VI 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 VII: Appendixes**
  Part VII includes several helpful quick-start guides for the Android development tools: the emulator, ADB and DDMS, Eclipse tips and tricks, and a SQLite tutorial.

**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, and it was available only in the United States. Today there are dozens of devices shipping all over the world. 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.

Within this new edition, we took the opportunity to do a serious overhaul on book content—but don’t worry, it’s still the book readers loved the first time, just bigger, better, and more comprehensive. In addition to adding newly available content, we’ve retested and upgraded all existing content (text and sample code) for use with the newest Android SDKs. 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
- Complete overhaul of sample code and applications—many more of them, too—organized by topic
- Nine new chapters, which cover new SDK features, including web APIs, the Android NDK, extending application reach, managing users, data synchronization, backups, advanced user input, and compatibility
- Topics such as Android Manifest files, content providers, designing apps, and testing each now have their own chapter
- Updated 3D graphics programming, including OpenGL ES 2.0
- Coverage of hot topics such as Bluetooth, gestures, voice recognition, App Widgets, Live Folders, Live Wallpapers, and global search
- 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
- A new appendix full of Eclipse tips and tricks

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 upon 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 10.6.4
- Eclipse Java IDE Version 3.5 (Galileo)
- Eclipse JDT plug-in and Web Tools Platform (WTP)
- Java SE Development Kit (JDK) 6 Update 20
Android SDK Version 2.2, API Level 8 (FroYo)
1. ADT Plug-in for Eclipse 0.9.9
2. NDK Tools Revision 4b
3. SDK Tools Revision 7

Android Handsets: T-Mobile G1, HTC Nexus One, HTC Evo 4G, Motorola Droid, ARCHOS 5 internet tablet

Supplementary Materials Available
The source code that accompanies this book for download on the publisher website:

We also run a blog at http://androidbook.blogspot.com, which covers a variety of Android topics and presents reader feedback, questions, and further information. You can also find links to our various technical articles.

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/

- **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/
**Introduction**

- **FierceDeveloper:** A weekly newsletter for wireless developers: http://www.fiercedeveloper.com/
- **Wireless Developer Network:** Daily news on the wireless industry: http://www.wirelessdevnet.com/
- **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.

This book also presents information in the following sidebars:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>Tips provide useful information or hints related to the current text.</td>
</tr>
<tr>
<td>Note</td>
<td>Notes provide additional information that might be interesting or relevant.</td>
</tr>
<tr>
<td>Warning</td>
<td>Warnings provide hints or tips about pitfalls that you might encounter and how to avoid them.</td>
</tr>
</tbody>
</table>

**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+awad2e@gmail.com
The mobile development community is at a tipping point. Mobile users demand more choice, more opportunities to customize their phones, and more functionality. Mobile operators want to provide value-added content to their subscribers in a manageable and lucrative way. Mobile developers want the freedom to develop the powerful mobile applications users demand with minimal roadblocks to success. Finally, handset manufacturers want a stable, secure, and affordable platform to power their devices. Up until now a single mobile platform has adequately addressed the needs of all the parties.

Enter Android, which is a potential game-changer for the mobile development community. An innovative and open platform, Android is well positioned to address the growing needs of the mobile marketplace.

This chapter explains what Android is, how and why it was developed, and where the platform fits in to the established mobile marketplace.

A Brief History of Mobile Software Development
To understand what makes Android so compelling, we must examine how mobile development has evolved and how Android differs from competing platforms.

Way Back When
Remember way back when a phone was just a phone? When we relied on fixed landlines? When we ran for the phone instead of pulling it out of our pocket? When we lost our friends at a crowded ballgame and waited around for hours hoping to reunite? When we forgot the grocery list (see Figure 1.1) and had to find a payphone or drive back home again?

Those days are long gone. Today, commonplace problems such as these are easily solved with a one-button speed dial or a simple text message like “WRU?” or “20?” or “Milk and?”

Our mobile phones keep us safe and connected. Now we roam around freely, relying on our phones not only to keep in touch with friends, family, and coworkers, but also to
Consider the following true story, which has been slightly enhanced for effect:

Once upon a time, on a warm summer evening, I was happily minding my own business cooking dinner in my new house in rural New Hampshire when a bat swooped over my head, scaring me to death.

The first thing I did—while ducking—was to pull out my cell phone and send a text message to my husband, who was across the country at the time. I typed, “There’s a bat in the house!”

My husband did not immediately respond (a divorce-worthy incident, I thought at the time), so I called my dad and asked him for suggestions on how to get rid of the bat.

He just laughed.
Annoyed, I snapped a picture of the bat with my phone and sent it to my husband and my blog, simultaneously guilt-tripping him and informing the world of my treacherous domestic wildlife encounter.

Finally, I googled “get rid of a bat” and then I followed the helpful do-it-yourself instructions provided on the Web for people in my situation. I also learned that late August is when baby bats often leave the roost for the first time and learn to fly. Newly aware that I had a baby bat on my hands, I calmly got a broom and managed to herd the bat out of the house.

Problem solved—and I did it all with the help of my trusty cell phone, the old LG VX9800.

My point here? Mobile phones can solve just about anything—and we rely on them for everything these days.

You notice that I used half a dozen different mobile applications over the course of this story. Each application was developed by a different company and had a different user interface. Some were well designed; others not so much. I paid for some of the applications, and others came on my phone.

As a user, I found the experience functional, but not terribly inspiring. As a mobile developer, I wished for an opportunity to create a more seamless and powerful application that could handle all I’d done and more. I wanted to build a better bat trap, if you will.

Before Android, mobile developers faced many roadblocks when it came to writing applications. Building the better application, the unique application, the competing application, the hybrid application, and incorporating many common tasks such as messaging and calling in a familiar way were often unrealistic goals.

To understand why, let’s take a brief look at the history of mobile software development.

“The Brick”

The Motorola DynaTAC 8000X was the first commercially available cell phone. First marketed in 1983, it was $13 \times 1.75 \times 3.5$ inches in dimension, weighed about 2.5 pounds, and allowed you to talk for a little more than half an hour. It retailed for $3,995, plus hefty monthly service fees and per-minute charges.

We called it “The Brick,” and the nickname stuck for many of those early mobile phones we alternatively loved and hated. About the size of a brick, with a battery power just long enough for half a conversation, these early mobile handsets were mostly seen in the hands of traveling business execs, security personnel, and the wealthy. First-generation mobile phones were just too expensive. The service charges alone would bankrupt the average person, especially when roaming.

Early mobile phones were not particularly full featured. (Although, even the Motorola DynaTAC, shown in Figure 1.2, had many of the buttons we’ve come to know well, such as the SEND, END, and CLR buttons.) These early phones did little more than make and receive calls and, if you were lucky, there was a simple contacts application that wasn’t impossible to use.
The first-generation mobile phones were designed and developed by the handset manufacturers. Competition was fierce and trade secrets were closely guarded. Manufacturers didn’t want to expose the internal workings of their handsets, so they usually developed the phone software in-house. As a developer, if you weren’t part of this inner circle, you had no opportunity to write applications for the phones.

It was during this period that we saw the first “time-waster” games begin to appear. Nokia was famous for putting the 1970s video game Snake on some of its earliest monochrome phones. Other manufacturers followed suit, adding games such as Pong, Tetris, and Tic-Tac-Toe.

These early phones were flawed, but they did something important—they changed the way people thought about communication. As mobile phone prices dropped, batteries improved, and reception areas grew, more and more people began carrying these handy devices. Soon mobile phones were more than just a novelty.

Customers began pushing for more features and more games. But there was a problem. The handset manufacturers didn’t have the motivation or the resources to build every application users wanted. They needed some way to provide a portal for entertainment and information services without allowing direct access to the handset.

What better way to provide these services than the Internet?
Wireless Application Protocol (WAP)

As it turned out, allowing direct phone access to the Internet didn’t scale well for mobile.

By this time, professional websites were full color and chock full of text, images, and other sorts of media. These sites relied on JavaScript, Flash, and other technologies to enhance the user experience, and they were often designed with a target resolution of 800x600 pixels and higher.

When the first clamshell phone, the Motorola StarTAC, was released in 1996, it merely had an LCD 10-digit segmented display. (Later models would add a dot-matrix type display.) Meanwhile, Nokia released one of the first slider phones, the 8110—fondly referred to as “The Matrix Phone” because the phone was heavily used in films. The 8110 could display four lines of text with 13 characters per line. Figure 1.3 shows some of the common phone form factors.

With their postage stamp-sized low-resolution screens and limited storage and processing power, these phones couldn’t handle the data-intensive operations required by traditional web browsers. The bandwidth requirements for data transmission were also costly to the user.

The Wireless Application Protocol (WAP) standard emerged to address these concerns. Simply put, WAP was a stripped-down version of HTTP, which is the backbone protocol of the Internet. Unlike traditional web browsers, WAP browsers were designed to run within the memory and bandwidth constraints of the phone. Third-party WAP sites
served up pages written in a markup language called Wireless Markup Language (WML). These pages were then displayed on the phone’s WAP browser. Users navigated as they would on the Web, but the pages were much simpler in design.

The WAP solution was great for handset manufacturers. The pressure was off—they could write one WAP browser to ship with the handset and rely on developers to come up with the content users wanted.

The WAP solution was great for mobile operators. They could provide a custom WAP portal, directing their subscribers to the content they wanted to provide, and rake in the data charges associated with browsing, which were often high.

Developers and content providers didn’t deliver. For the first time, developers had a chance to develop content for phone users, and some did so, with limited success.

Most of the early WAP sites were extensions of popular branded websites, such as CNN.com and ESPN.com, which were looking for new ways to extend their readership. Suddenly phone users accessed the news, stock market quotes, and sports scores on their phones.

Commercializing WAP applications was difficult, and there was no built-in billing mechanism. Some of the most popular commercial WAP applications that emerged during this time were simple wallpaper and ringtone catalogues that enabled users to personalize their phones for the first time. For example, a user browsed a WAP site and requested a specific item. He filled out a simple order form with his phone number and his handset model. It was up to the content provider to deliver an image or audio file compatible with the given phone. Payment and verification were handled through various premium-priced delivery mechanisms such as Short Message Service (SMS), Enhanced Messaging Service (EMS), Multimedia Messaging Service (MMS), and WAP Push.

WAP browsers, especially in the early days, were slow and frustrating. Typing long URLs with the numeric keypad was onerous. WAP pages were often difficult to navigate. Most WAP sites were written one time for all phones and did not account for individual phone specifications. It didn’t matter if the end user’s phone had a big color screen or a postage stamp-sized monochrome screen; the developer couldn’t tailor the user’s experience. The result was a mediocre and not very compelling experience for everyone involved.

Content providers often didn’t bother with a WAP site and instead just advertised SMS short codes on TV and in magazines. In this case, the user sent a premium SMS message with a request for a specific wallpaper or ringtone, and the content provider sent it back. Mobile operators generally liked these delivery mechanisms because they received a large portion of each messaging fee.

WAP fell short of commercial expectations. In some markets, such as Japan, it flourished, whereas in others, such as the United States, it failed to take off. Handset screens were too small for surfing. Reading a sentence fragment at a time, and then waiting seconds for the next segment to download, ruined the user experience, especially because every second of downloading was often charged to the user. Critics began to call WAP “Wait and Pay.”
Finally, the mobile operators who provided the WAP portal (the default home page loaded when you started your WAP browser) often restricted which WAP sites were accessible. The portal enabled the operator to restrict the number of sites users could browse and to funnel subscribers to the operator’s preferred content providers and exclude competing sites. This kind of walled garden approach further discouraged third-party developers, who already faced difficulties in monetizing applications, from writing applications.

Proprietary Mobile Platforms

It came as no surprise that users wanted more—they will always want more.

Writing robust applications with WAP, such as graphic-intensive video games, was nearly impossible. The 18-year-old to 25-year-old sweet-spot demographic—the kids with the disposable income most likely to personalize their phones with wallpapers and ringtones—looked at their portable gaming systems and asked for a device that was both a phone and a gaming device or a phone and a music player. They argued that if devices such as Nintendo’s Game Boy could provide hours of entertainment with only five buttons, why not just add phone capabilities? Others looked to their digital cameras, Palms, BlackBerries, iPods, and even their laptops and asked the same question. The market seemed to be teetering on the edge of device convergence.

Memory was getting cheaper, batteries were getting better, and PDAs and other embedded devices were beginning to run compact versions of common operating systems such as Linux and Windows. The traditional desktop application developer was suddenly a player in the embedded device market, especially with smartphone technologies such as Windows Mobile, which they found familiar.

Handset manufacturers realized that if they wanted to continue to sell traditional handsets, they needed to change their protectionist policies pertaining to handset design and expose their internal frameworks to some extent.

A variety of different proprietary platforms emerged—and developers are still actively creating applications for them. Some smartphone devices ran Palm OS (now Garnet OS) and RIM BlackBerry OS. Sun Microsystems took its popular Java platform and J2ME emerged (now known as Java Micro Edition [Java ME]). Chipset maker Qualcomm developed and licensed its Binary Runtime Environment for Wireless (BREW). Other platforms, such as Symbian OS, were developed by handset manufacturers such as Nokia, Sony Ericsson, Motorola, and Samsung. The Apple iPhone OS (OS X iPhone) joined the ranks in 2008. Figure 1.4 shows several different phones, all of which have different development platforms.

Many of these platforms have associated developer programs. These programs keep the developer communities small, vetted, and under contractual agreements on what they can and cannot do. These programs are often required and developers must pay for them.

Each platform has benefits and drawbacks. Of course, developers love to debate about which platform is “the best.” (Hint: It’s usually the platform we’re currently developing for.)

The truth is that no one platform has emerged victorious. Some platforms are best suited for commercializing games and making millions—if your company has brand
backing. Other platforms are more open and suitable for the hobbyist or vertical market applications. No mobile platform is best suited for all possible applications. As a result, the mobile phone has become increasingly fragmented, with all platforms sharing part of the pie.

For manufacturers and mobile operators, handset product lines quickly became complicated. Platform market penetration varies greatly by region and user demographic. Instead of choosing just one platform, manufacturers and operators have been forced to sell phones for all the different platforms to compete in the market. We’ve even seen some handsets supporting multiple platforms. (For instance, Symbian phones often also support J2ME.)

The mobile developer community has become as fragmented as the market. It’s nearly impossible to keep track of all the changes in the market. Developer specialty niches have formed. The platform development requirements vary greatly. Mobile software developers work with distinctly different programming environments, different tools, and different programming languages. Porting among the platforms is often costly and not straightforward. Keeping track of handset configurations and testing requirements, signing and certification programs, carrier relationships, and application marketplaces have become complex spin-off businesses of their own.
It’s a nightmare for the ACME Company that wants a mobile application. Should it develop a J2ME application? BREW? iPhone? Windows Mobile? Everyone has a different kind of phone. ACME is forced to choose one or, worse, all of the platforms. Some platforms allow for free applications, whereas others do not. Vertical market application opportunities are limited and expensive.

As a result, many wonderful applications have not reached their desired users, and many other great ideas have not been developed at all.

The Open Handset Alliance

Enter search advertising giant Google. Now a household name, Google has shown an interest in spreading its vision, its brand, its search and ad-revenue-based platform, and its suite of tools to the wireless marketplace. The company’s business model has been amazingly successful on the Internet and, technically speaking, wireless isn’t that different.

Google Goes Wireless

The company’s initial forays into mobile were beset with all the problems you would expect. The freedoms Internet users enjoyed were not shared by mobile phone subscribers. Internet users can choose from the wide variety of computer brands, operating systems, Internet service providers, and web browser applications.

Nearly all Google services are free and ad driven. Many applications in the Google Labs suite directly compete with the applications available on mobile phones. The applications range from simple calendars and calculators to navigation with Google Maps and the latest tailored news from News Alerts—not to mention corporate acquisitions such as Blogger and YouTube.

When this approach didn’t yield the intended results, Google decided to a different approach—to revamp the entire system upon which wireless application development was based, hoping to provide a more open environment for users and developers: the Internet model. The Internet model allows users to choose between freeware, shareware, and paid software. This enables free market competition among services.

Forming the Open Handset Alliance

With its user-centric, democratic design philosophies, Google has led a movement to turn the existing closely guarded wireless market into one where phone users can move between carriers easily and have unfettered access to applications and services. With its vast resources, Google has taken a broad approach, examining the wireless infrastructure from the FCC wireless spectrum policies to the handset manufacturers’ requirements, application developer needs, and mobile operator desires.

Next, Google joined with other like-minded members in the wireless community and posed the following question: What would it take to build a better mobile phone?

The Open Handset Alliance (OHA) was formed in November 2007 to answer that very question. The OHA is a business alliance comprised of many of the largest and most
successful mobile companies on the planet. Its members include chip makers, handset manufacturers, software developers, and service providers. The entire mobile supply chain is well represented.

Andy Rubin has been credited as the father of the Android platform. His company, Android Inc., was acquired by Google in 2005. Working together, OHA members, including Google, began developing a nonproprietary open standard platform based upon technology developed at Android Inc. that would aim to alleviate the aforementioned problems hindering the mobile community. The result is the Android project. To this day, most Android platform development is completed by Rubin’s team at Google, where he acts as VP of Engineering and manages the Android platform roadmap.

Google’s involvement in the Android project has been so extensive that the line between who takes responsibility for the Android platform (the OHA or Google) has blurred. Google hosts the Android open source project and provides online Android documentation, tools, forums, and the Software Development Kit (SDK) for developers. All major Android news originates at Google. The company has also hosted a number of events at conferences and the Android Developer Challenge (ADC), a contest to encourage developers to write killer Android applications—for $10 million dollars in prizes to spur development on the platform. The winners and their apps are listed on the Android website.

**Manufacturers: Designing the Android Handsets**

More than half the members of the OHA are handset manufacturers, such as Samsung, Motorola, HTC, and LG, and semiconductor companies, such as Intel, Texas Instruments, NVIDIA, and Qualcomm. These companies are helping design the first generation of Android handsets.

The first shipping Android handset—the T-Mobile G1—was developed by handset manufacturer HTC with service provided by T-Mobile. It was released in October 2008. Many other Android handsets were slated for 2009 and early 2010. The platform gained momentum relatively quickly. Each new Android device was more powerful and exciting than the last. Over the following 18 months, 60 different Android handsets (made by 21 different manufacturers) debuted across 59 carriers in 48 countries around the world. By June 2010, at an announcement of a new, highly anticipated Android handset, Google announced more than 160,000 Android devices were being activated each day (for a rate of nearly 60 million devices annually). The advantages of widespread manufacturer and carrier support appear to be really paying off at this point.

The Android platform is now considered a success. It has shaken the mobile marketplace, gaining ground steadily against competitive platforms such as the Apple iPhone, RIM BlackBerry, and Windows Mobile. The latest numbers (as of Summer 2010) show BlackBerry in the lead with a declining 31% of the smartphone market. Trailing close behind is Apple’s iPhone at 28%. Android, however, is trailing with 19%, though it’s gaining ground rapidly and, according to some sources, is the fastest-selling smartphone platform. Microsoft Windows Mobile has been declining and now trails Android by several percentage points.
Mobile Operators: Delivering the Android Experience
After you have the phones, you have to get them out to the users. Mobile operators from North, South, and Central America; Europe, Asia, India, Australia, Africa, and the Middle East have joined the OHA, ensuring a worldwide market for the Android movement. With almost half a billion subscribers alone, telephony giant China Mobile is a founding member of the alliance.

Much of Android’s success is also due to the fact that many Android handsets don’t come with the traditional “smartphone price tag”—quite a few are offered free with activation by carriers. Competitors such as the Apple iPhone have no such offering as of yet. For the first time, the average Jane or Joe can afford a feature-full phone. I’ve lost count of the number of times I’ve had a waitress, hotel night manager, or grocery store checkout person tell me that they just got an Android phone and it has changed their life. This phenomenon has only added to the Android’s rising underdog status.

In the United States, the Android platform was given a healthy dose of help from carriers such as Verizon, who launched a $100 million dollar campaign for the first Droid handset. Many other Droid-style phones have followed from other carriers. Sprint recently launched the Evo 4G (America’s first 4G phone) to much fanfare and record one-day sales (http://j.mp/cNhbb4b).

Content Providers: Developing Android Applications
When users have Android handsets, they need those killer apps, right?

Google has led the pack, developing Android applications, many of which, such as the email client and web browser, are core features of the platform. OHA members are also working on Android application integration. eBay, for example, is working on integration with its online auctions.

The first ADC received 1,788 submissions, with the second ADC being voted upon by 26,000 Android users to pick a final 200 applications that would be judged professionally—all newly developed Android games, productivity helpers, and a slew of location-based services (LBS) applications. We also saw humanitarian, social networking, and mash-up apps. Many of these applications have debuted with users through the Android Market—Google’s software distribution mechanism for Android. For now, these challenges are over. The results, though, are still impressive.

For those working on the Android platform from the beginning, handsets couldn’t come fast enough. The T-Mobile G1 was the first commercial Android device on the market, but it had the air of a developer pre-release handset. Subsequent Android handsets have had much more impressive hardware, allowing developers to dive in and design awesome new applications.
As of October 2010, there are more than 80,000 applications available in the Android Market, which is growing rapidly. This takes into account only applications published through this one marketplace—not the many other applications sold individually or on other markets. This also does not take into account that, as of Android 2.2, Flash applications can run on Android handsets. This opens up even more application choices for Android users and more opportunities for Android developers.

There are now more than 180,000 Android developers writing interesting and exciting applications. By the time you finish reading this book, you will be adding your expertise to this number.

Taking Advantage of All Android Has to Offer

Android’s open platform has been embraced by much of the mobile development community—extending far beyond the members of the OHA.

As Android phones and applications have become more readily available, many other mobile operators and handset manufacturers have jumped at the chance to sell Android phones to their subscribers, especially given the cost benefits compared to proprietary platforms. The open standard of the Android platform has resulted in reduced operator costs in licensing and royalties, and we are now seeing a migration to open handsets from proprietary platforms such as RIM, Windows Mobile, and the Apple iPhone. The market has cracked wide open; new types of users are able to consider smartphones for the first time. Android is well suited to fill this demand.

Android Platform Differences

Android is hailed as “the first complete, open, and free mobile platform”:

- **Complete**: The designers took a comprehensive approach when they developed the Android platform. They began with a secure operating system and built a robust software framework on top that allows for rich application development opportunities.

- **Open**: The Android platform is provided through open source licensing. Developers have unprecedented access to the handset features when developing applications.

- **Free**: Android applications are free to develop. There are no licensing or royalty fees to develop on the platform. No required membership fees. No required testing fees. No required signing or certification fees. Android applications can be distributed and commercialized in a variety of ways.

Android: A Next-Generation Platform

Although Android has many innovative features not available in existing mobile platforms, its designers also leveraged many tried-and-true approaches proven to work in the wireless world. It’s true that many of these features appear in existing proprietary
platforms, but Android combines them in a free and open fashion while simultaneously addressing many of the flaws on these competing platforms.

The Android mascot is a little green robot, shown in Figure 1.5. This little guy (girl?) is often used to depict Android–related materials.

Android is the first in a new generation of mobile platforms, giving its platform developers a distinct edge on the competition. Android’s designers examined the benefits and drawbacks of existing platforms and then incorporated their most successful features. At the same time, Android’s designers avoided the mistakes others suffered in the past.

Since the Android 1.0 SDK was released, Android platform development has continued at a fast and furious pace. For quite some time, there was a new Android SDK out every couple of months! In typical tech-sector jargon, each Android SDK has had a project name. In Android’s case, the SDKs are named alphabetically after sweets (see Figure 1.6).

The latest version of Android is codenamed Gingerbread.
Free and Open Source
Android is an open source platform. Neither developers nor handset manufacturers pay royalties or license fees to develop for the platform.

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

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

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

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

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

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

Reasonable Learning Curve for Developers
Android applications are written in a well-respected programming language: Java.

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

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

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

Rich, Secure Application Integration

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

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

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

No Costly Obstacles to Publication

Android applications have none of the costly and time-intensive testing and certification programs required by other platforms such as BREW and Symbian.
A “Free Market” for Applications

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

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

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

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

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

  ![Figure 1.7 The Android market.](image)

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

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

A New and Growing Platform

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

On the other hand, developers diving into Android development now benefit from the first-to-market competitive advantages we’ve seen on other platforms such as BREW
Android is an operating system and a software platform upon which applications are developed. A core set of applications for everyday tasks, such as web browsing and email, are included on Android handsets.

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

Android’s Underlying Architecture

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

The Linux Operating System

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

- Enforcement of application permissions and security
- Low-level memory management
- Process management and threading
- The network stack
- Display, keypad input, camera, Wi-Fi, Flash memory, audio, and binder (IPC) driver access

Figure 1.8 Diagram of the Android platform architecture.
**Android Application Runtime Environment**
Each Android application runs in a separate process, with its own instance of the Dalvik virtual machine (VM). Based on the Java VM, the Dalvik design has been optimized for mobile devices. The Dalvik VM has a small memory footprint, and multiple instances of the Dalvik VM can run concurrently on the handset.

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

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

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

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

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

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

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

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

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

Let’s say we have an application that keeps track of the user’s public and private birthday wish lists. If this application wanted to share its data with other applications, it could grant URI permissions for the public wish list, allowing another application permission to access this list without explicitly having to ask for it.
Chapter 1  Introducing Android

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

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

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

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

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

Figure 1.9  Duke, the Java mascot.
No Distinctions Made Between Native and Third-Party Applications

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

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

Commonly Used Packages

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

The Android packages include support for

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

Android Application Framework

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

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

Android applications can interact with the operating system and underlying hardware using a collection of managers. Each manager is responsible for keeping the state of some underlying system service. For example, there is a LocationManager that facilitates interaction with the location-based services available on the handset. The WindowManager and WindowManager manage user interface fundamentals.
Applications can interact with one another by using or acting as a **ContentProvider**. Built-in applications such as the Contact manager are content providers, allowing third-party applications to access contact data and use it in an infinite number of ways. The sky is the limit.

**Summary**

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

**References and More Information**

Android Development:
   - [http://developer.android.com](http://developer.android.com)
Open Handset Alliance:
   - [http://www.openhandsetalliance.com](http://www.openhandsetalliance.com)
Index

Symbols

# (hash symbol), 111
… (ellipsis), 136

3D graphics

cubes, drawing, 378
lighting, 379-382
OpenGL ES, 368
SurfaceView, creating, 370
texturing, 381-384
vertices
  coloring, 377-378
drawing, 376-377

3GPP Specifications website, 365

A

AbsoluteLayout class, 190
AbstractAccountAuthenticator class, 490
abstracted LCD density AVD hardware option, , 620
AbstractThreadedSyncAdapter class, 491
AccelerateDecelerateInterpolator, 230
AccelerateInterpolator, 230
accelerometer sensor, 410-411, 619
accessibility framework, 502-503
  android.speech package, 503
  speech recognition services, 504-506
Text-To-Speech services, 503, 506-508
  converting text into sound files, 508
  initializing, 507
language settings, 507
OnInitListener interface, 506
accessing
application preferences, 70
Browser content provider, 263
Contacts private data, 264-266
ccontent providers with permissions, 262-263
database files, 240
device sensors, 408-409
hardware, 407
images, 270-271
Internet. See HTTP
layout XML, 126
menus, 120
preferences, 231-234
resources, 103
secondary logs, 654
strings, 108-109
telephony state, 354
WiFi networks, 412-413
AccountManager class, 490, 497
accounts
AccountManager class, 490
android.accounts package, 489
authenticators, 490
credentials, protecting, 490
developer accounts
benefits, 609
creating, 604-606
Distribution Agreement, 604
providers, 490
registering, 490
sync adapters, 491
activities
App Widget configuration, 455
dialogs, adding, 166-167
external, launching, 77
game application examples, 71
intents, processing, 468
lifecycle, 72
callbacks, 72-73
destroying activities, 75
initializing static activity data, 74
killing activities, 75
releasing activity data, 74
retrieving activity data, 74
saving activity data, 74
saving state to Bundle objects, 75
stopping activity data, 74
live folders, 282, 481-482
manifest file definition, 92
MapActivity, 324
organizing with menus, 78
primary entry point, 92-93
reference website, 80
searches, creating, 475-477
stacks, 72
starting, 76-77
themes, applying, 170
transitioning with intents, 76
action/data types, 77
external Activities, launching, 77
new activities, launching, 76-77
passing additional information, 78
Activity class, 71
<activity> tag, 92
ad revenue, 612
adapters, 194
arrays, 194-195
binding data, 196
cursor, 195-196
database data, binding, 254-256
event handling, 197
ImageUriAdapter, 272
sync, 491, 497
AdapterView classes
ArrayAdapter class, 194-195
binding data, 196
CursorAdapter class, 195-196
event handling, 197

ADB (Android Debug Bridge), 39
applications
installing, 651
reinstalling, 651
testing, 656
uninstalling, 651
backup services
archived data, wiping, 655
controlling, 654-655
forcing restores, 655
scheduling, 655
bug reports, 655-656
command listing, 660
connected devices/emulators, listing, 647-648
copying files, 650
custom binaries, installing, 659-660
functionality, 647
LogCat utility
clearing logs, 654
dates and times, 652
filtering, 652-653
output redirection, 654
secondary logs, accessing, 654
viewing logs, 652
shell commands, 649-650
emulator, starting/stopping, 649-650
issuing single, 649
shell sessions, starting, 649
specific device commands, 648
sqlite3 database tool, 656
starting/stopping server processes, 648
stress testing applications
event listening, 656-657
event types, weighting, 657-658
monkey tool, launching, 656
repeating events, 658
throttle, 658
website, 39

ADC (Android Developer Challenge), 16
addGlobalFocusChangeListener() method, 163
addGlobalLayoutChangeLister() method, 163
addOnPreDrawListener() method, 163
addOnTouchListener() method, 162
addView() method, 178
ad-hoc permissions, 25
Adobe AIR
applications, building, 313
beta program, 313
Tool Suite website, 314

ADT plug-in, 35-36
AIDL (Android Interface Definition Language)
Parcelable class file, 448
remote interfaces, declaring, 444
alert dialogs, 165
aliases (resources), 123
alpha transparency transformations, 228
alternate marketplaces, 610-611
alternative layouts, 127
alternative resources, 102-103, 531
configuration changes, handling, 539
data retention, 539
default application icon resources example, 531
directory qualifiers
Android platform, 536
applying, 532
bad examples, 536-537
case, 532
combining, 532
default resources, 536
dock mode, 534
good examples, 536
keyboard type and availability, 535
language and region code, 533
mobile country code, 533
mobile network code, 533
names, 532
navigation key availability, 535
navigation method, 536
night mode, 534
required strings, 533
screen aspect ratio, 534
screen orientation, 534
screen pixel density, 534
screen size, 533
text input method, 535
touch screen type, 535
efficiency, 538-539
hierarchy, 531
internationalization, 540-542
device language and locale example, 541-542
dual language support example, 540-541
performance, 539
programmatic configurations, 538
screen orientation customization example, 537-538
websites, 549

AnalogClock class, 156-157

Android

benefits, 18
completeness, 18

Debug Bridge. See ADB
Dev Guide: “Developing on a Device” website, 67
Developer Challenge (ADC), 16
Developers blog, 574
Development website, 28, 398
first-to-market advantages, 23
freedoms, 18
Interface Definition Language. See AIDL
mascot/logo, 19
open source, 18, 20
packages, 35, 131
android.accounts, 489
android.bluetooth, 415
android.content, 232
android.database.sqlite, 239
android.gesture, 509
android.graphics, 230
android.graphics.drawable.shapes, 215
android.hardware, 408, 412
android.sax.*, 237
android.speech, 503
android.telephony, 354, 357
android.test, 582
android.util.Xml.*, 237
android.view, 133
android.view.animation, 226
android.webkit, 307
android.widget, 134
Project Wizard, 44
Virtual Devices. See AVDs

Android Market, 603-609
applications
deleting, 609
upgrading, 609
uploading, 606-608
country requirements, 604
developer accounts
  benefits, 609
  creating, 604-606
  Distribution Agreement, 604
help, 607
licensing service, 604
publication, 608
refund policy, 608-609
sign-up website, 604
website, 612

AndroidManifest.xml file, 52
Android.net package website, 299

animations, 116
  android.view.animation package, 226
  frame-by-frame, 116, 117, 223-225
    animation loops, naming, 224
    genie juggling gifts example, 223-224
    starting, 224
    stopping, 224
  helper utilities, 116
  interpolators, 230
  loading, 227-228
  moving, 229-230
  rotating, 228-229
  scaling, 229
  storing, 101
  transparency, 228
tweening, 116-118, 224-230
  defining as XML resources, 226
  defining programmatically, 226
  loading, 227-228
  moving transformations, 229-230
  rotating, 228-229
  scaling, 229
  simultaneously/sequentially, 226-227
  transformations, defining, 224
  transparency, 228
types, 221-223

AnimationUtils class, 227-228
antialiasing paints, 207
AnticipateInterpolator, 230
AnticipateOvershootInterpolator, 230
Apache Software License (ASL/Apache2), 20
API levels
  finding, 546-547
  website, 96

ApiDemos application, 40

App Widgets
  AppWidgetProvider class, 455
    creating
      application support, 453
      configuration activities, 455
      dimensions, 454
      providers, 455
      sizing, 454
      XML definition, 453-454
    hosts, 460
    implementing, 455-456
    installing, 460-461
    manifest file, configuring, 459
    overview, 452-453
    providers, 452
    reference websites, 487
    update service, creating, 458-459
    updating, 453, 454
      onUpdate() method, 458
    view hierarchies, 456-457

applications
  activities. See activities
Adobe AIR, building, 313
ApiDemos, 40
architectures, 565
AVDs, creating, 51
Browser, 302
build targets, 50
compatibility
alternative resources. See alternative resources
device differentiators, 523-524
forward, 554
hardware configuration support, 545-546
internationalization, 539-545
maximizing, 523-525
user interfaces, 525-531
versions, 546-548
website, 549
as content providers, 274
content provider interfaces, implementing, 275
data, adding, 278-279
data columns, defining, 276
deleting data, 280-281
manifest files, updating, 282
MIME types, returning, 281-282
queries, 276-277
updates, 279-280
URIs, 276-277
as content type handlers, 466-467
Context, 70
Activity instances, 71
application preferences, accessing, 70
application resources, retrieving, 70
retrieving, 70
core files/directories, 52-51
debugging
emulator, 56-59
on handsets, 65-66
registering as debuggable, 65
developing, 568
descriptions, 87
developer competition, 21
development. See development
distributing
ad revenue, 612
alternate marketplaces, 610-611
Android Market, 603-609
billing users, 611-612
considerations, 597-598
copy protection, 611
manufacturer/operator partnerships, 611
self-distribution, 609-610
enhancing, 451-452
extending, 451-452
files, backing up, 494-495
Flash
Adobe AIR applications, building, 313
advantages/disadvantages, 311-312
enabling, 312-313
framework, 27-28
free market, 22
functionality, 97
global searches, enabling, 478
heap activity, monitoring, 639-640
hello-jni sample, 399
icons, 87
images, adding, 269
accessing images, 270-271
binding data to Gallery control, 272
data retrieval, 272
finding content with URIs, 271
gallery image retrieval, 273
retrieved images, viewing, 273-274
implementing, 567
installing, 593, 651
integration, 21-12
interoperability, 451
JavaScript interface application, 308-312
  Button control click handler, 311
  JavaScript control, 311
  JavaScript namespace, 309
  JavaScriptExtensions class, 309
  onCreate() method, 309
  sample.html file JavaScript functions, 310-311
  web page, defining, 310
launch configurations, creating, 52-53
location-based services, adding, 62-64
  AVDs with Google APIs, creating, 62
  emulator location, configuring, 62-63
  last known location, finding, 63-64
logging, adding, 59-60
LunarLander, 40
media, adding, 60-62
names, 50, 87
network-driven, 565
NotePad, 40
as operating system users, 25
packaging preparations
  debugging, disabling, 600
  icons, 599
  logging, disabling, 600
  manifest files for market filtering, configuring, 599
  market requirements, 599-600
  names, 50, 599
  permissions, 600
target platforms, verifying, 599
versions, 599
permissions, 25
PetTracker
  binding data, 253-244
  field names, 251
  SQLiteOpenHelper class, extending, 251-256
PetTracker3, 270-274
preferences
  accessing, 70, 231-234
  adding, 232, 233-234
  data types, 231
  deleting, 233
  file formats, 234
  finding, 232
  functionality, 232
  methods, 233
  private, 232
  reading, 232
  shared, 232
  updating, 234
projects, creating, 50
publishing
  Android Market, 608
  certification, 603
  exporting package files, 601-602
  release versions, testing, 603
  requirements, 598
  signing package files, 600-602
reinstalling, 651
resources
  accessing programmatically, 103
  adding, 98
  aliases, 123
  alternative, 102-103
  animations. See animations
Boolean, 110
colors. See colors
default, 132
defined, 97
defining types with Eclipse, 104-107
dimensions, 112-113
directory hierarchy, 97-98
drawables, 113-114
images. See images
integer, 111
layout. See layouts
menus. See menus
raw files, 121-122
referencing, 122-123
retrieving, 70
selector, 116
storing, 97, 101
strings. See strings
styles, 127-130
system, referencing, 131
themes, 131
types, 99-101
website, 132
XML files, 120-121
responsiveness, 573-574
running in Android emulator, 47-48, 53-55
sample, 40
screen orientation customization example, 537-538
searches, 469-470-471
activities, creating, 475-477
enabling, 471-472
manifest files, 477-478
Search buttons, 478
Searchable Configuration documentation website, 475
suggestions, 472-474
voice capabilities, 474-475
website, 487
XML configuration file, 471
SimpleDatabase
file, accessing, 240
openOrCreateDatabase() method, 240
properties, configuring, 241
SimpleMultiTouchGesture example, 516-519
SimpleNDK, 399-400
exception handling, 402-403
parameters, handling, 401-402
return values, handling, 401-402
Snake, 40
adding to Eclipse workspace, 43-44
AVD, creating, 44-46
launch configurations, creating, 46-48
running in Android emulator, 47-48
stability, 573-574
stand-alone, 565
support requirements, 568
documentation, 569
firmware upgrades, 569
live server changes, 569
low-risk porting, identifying, 569
user crash/bug reports, 569
Sync Adapter example, 491
testing, 567-568
threads
activity, monitoring, 638-639
viewing, 637-638
uninstalling, 651
uploading applications to Android Market, 606-608
user interfaces. See user interfaces versioning, 86
AppWidgetProvider class, 455
<appwidget-provider> tag, 454
architectures
  applications, 565
  platform, 23
    Linux Operating System, 23–24
  runtime environment, 25
arcs, drawing, 219-220
ArcShape object, 220
ArrayAdapter class, 194-195
arrays
  adapters, 194-195
  converting to buffers, 377
  strings, 109-110
ash shell, 649
ASL/Apache2 (Apache Software License), 20
Asset Packaging tool, 98
assets folder, 52
asynchronous processing, 291-293
AsyncTask class, 292-293
attributes
  autoLink, 136-137
  completionThreshold, 141
  ellipsize, 136
  ems, 136
  FrameLayout views, 183-185
  glEsVersion, 368
  hint, 138
  includeInGlobalSearch, 478
  inputType, 501
  interpolator, 230
  layouts, 181-182
  LinearLayout views, 186
  lines, 138
  maxEms, 136
  maxLines, 136
  maxSdkVersion, 88
  minEms, 136
  minLines, 136
  minSdkVersion, 88
  permission, 95
  prompt, 144
  RelativeLayout views, 187-189
  search suggestions, 472
  TableLayout views, 191
  targetSdkVersion, 88, 89
  textOn/textOff, 147
  TextView class, 135
  View class, 127
  ViewGroups, 182
audio, 346
  AudioManager service, 349
  finding, 350
  formats website, 351
  notifications, 431-432
  playing, 348-349, 620
  recording, 347-348, 619
  ringtones, 351
  sharing, 349-350
  voice searches, 474-475
  website, 351
Audio.Albums class, 260
Audio.Artists class, 260
Audio.Genres class, 260
AudioManager service, 349
Audio.Media class, 260
Audio.Playlists class, 260
audioRecorder object, 348
authenticators (accounts), 490
auto-complete
  Java code, 664
  text editors, 139-142
AutoCompleteTextView class, 139
autoLink attribute, 136-137
automated testing, 590
AVDs (Android Virtual Devices)
creating, 44-46, 51, 616-618
emulator
configuring, 616-617
launching, 623
phone call simulation, 625
settings, configuring, 615-616
Google APIs, 319
hardware options, 618-620
Manager, 36-37
skin options, 618
backup agents
implementing, 492-493
registering, 495-496
backup services, 491
application files, 494-495
archived data, wiping, 655
backup agents
implementing, 492-493
registering, 495-496
controlling with ADB, 654-655
forcing restores, 655
remote, choosing, 492
requesting backups, 496
restore operations, 496-497
scheduling, 655
shared preferences files, 493-494
troubleshooting, 497
website, 497
BackupAgentHelper class, 492
backward compatibility
Java Reflection, 547-548
without reflection website, 548
BasicGLThread class, 372
batteries
AVD hardware option, 619
monitoring, 417-420
BatteryManager class, 419
beginTransaction() method, 244
benefits, 18
best practices
design
Android Developers blog, 574
billing and revenue generation, 575
network diagnostics, 576-577
responsiveness, 573-574
rules, 571-572
silly mistakes, avoiding, 578
stability, 573-574
third-party standards, 576
tools, 578
updates/upgrades, 577-578
user demands, meeting, 572
user interfaces, 572-573
development, 579
code diagnostics, 581
code quality, 580
code reviews, 581
coding standards, 580-581
device specific bugs, 582
feasibility testing, 579-580
silly mistakes, avoiding, 583
software processes, 579
tools, 583
unit testing, 581-582
emulator, 613-614
security, 574
handling private data, 575
transmitting private data, 575
testing, 585
  application installations, 593
  automation, 590
  backup services, 594
  billing, 594
  black box, 591
  build acceptance tests, 589
  conformance, 593
  coverage, maximizing, 589
  defect tracking systems, 585-587
  device fragmentation, 587
  emulator limitations, 590-591
  emulator versus actual device, 589-590
  environments, 587
  integration points, 592-593
  internationalization, 593
  outsourcing, 596
  performance, 594
  preproduction devices, 590
  priorities, 588
  quality, 594
  real-life device configurations, 588
  servers, 591-592
  services, 591-592
  silly mistakes, avoiding, 595
  signal loss, 589
  software integration, 588-589
  specialized scenarios, 592
  starting states, 588
  third-party firmware, 587
  third-party standards, 592
  tools, 595
  unexpected events, 594
  upgrades, 593
  usability, 592
  white box, 591
  websites, 584

billing users, 611-612
  generation methods, 575
  testing, 594
bindService() method, 438
Bitmap class, 212
BitmapDrawable class, 116
bitmaps, 212
  Bitmap class, 212
  drawing, 213
  scaling, 213
  transforming into matrixes, 213
black box testing, 591
blinking light notifications, 430-431
  clearing, 431
  colors, 430
  customizing, 431
  precedence, 430
  testing, 430
  urgency, 430-431
Bluetooth
  available, finding, 415
  classes, 415
  connections, 416-417
  device discovery, 416
  enabling, 415-416
  functionality, 414
  implementation example, 417-418
  paired devices, querying, 416
  permissions, 415
  websites, 421
BluetoothAdapter class, 415
BluetoothDevice class, 415
BluetoothServerSocket class, 415
BluetoothSocket class, 415
Bodlaender, Hans, 211
bold strings, 108
<bool> tag, 110
Boolean resources, 110
Borland SilkTest, 589
BounceInterpolator, 230
boundCenterBottom() method, 330
broadcasting
  intents, 79
  receivers, registering, 93-94
broadcastIntent() method, 79
Browser application, 259, 263, 302
  accessing, 263
  querying for most visited bookmarked sites, 263-264
browser images, downloading, 271
browsing the Web, 301-302
  chrome, adding, 305-307
  event handling, 304-305
Flash support
  Adobe AIR applications, building, 313
  advantages/disadvantages, 311-312
  enabling, 312-313
JavaScript, enabling, 304
  mouseovers, 304
settings, configuring, 304
WebKit rendering engine, 301
  android.webkit package, 307
  classes, 307
  functionality, 308
JavaScript interface application, 308-312
  Open Source Project website, 314
  support, 307
zooming, 304
buffers, 377
bugs
  device specific, 582
  reports, 569, 655-656
  resolution process website, 32
build acceptance tests, 589
build errors, resolving, 667
build targets, 50
built-ins
  content providers, 259
  layouts, 181
  themes, 171
  view containers, 193
Bundle objects, 75
BusyBox
  binary, installing, 660
  websites, 660
Button class, 144
buttons, 144
  basic, 144-146
  check boxes, 144, 146-147
  images, 146
  margin example, 183
  radio, 144, 148-149
  Search, 478
  toggles, 144, 147
C
C2DM (Cloud to Device Messaging), 438
cache files
  AVD hardware option, 620
  creating, 238-239
  retrieving, 236
CacheManager class, 307
calculateAndDisplayFPS() method, 385
calibrating device sensors, 410-411
call states
  listening for changes, 355
  permissions, 354
  querying, 354-355
  roaming, 356
  service state, 355-356
CallLog content provider, 259, 261-263
access permissions, 262-263
tagging phone numbers with custom labels, 262
camera
AVD hardware option, 619
image capturing, 336-340
  adding, 336-337
  button click handler, 340
  Camera object, instantiating, 337-338
camera parameters, 338
  CameraSurfaceView class, 337
layouts, 339
starting preview, 338-339
stopping preview, 338
takePicture() method, 339
settings, configuring, 340-341
zoom controls, 341
Camera class, 340
CameraSurfaceView class, 337
cancel() method, 428
cancelDiscovery() method, 416
canDetectOrientation() method, 521
canvases, 205-207
  bitmaps, drawing, 213
  Canvas object, 207
dimensions, 207
  red circle on black canvas example, 205-206
cellular networks, emulating, 298
certifying applications, 603
change listeners
call states, 355
  entire screen events
    GlobalFocusChange events, 163
    GlobalLayout events, 163
    PreDraw events, 163
  focus changes, 164-165
touch mode changes, 161-162
character picker dialogs, 165
check boxes, 144, 146-147
chess font, 211
child views, 178
choosing
  build targets, 50
devices for device databases, 556
IDEs (integrated development environments), 20
paint colors, 207
programming languages, 26
remote backup services, 492
SDK versions
  maximum, 90
  minimum, 89
target, 89
software keyboards, 500-502
source control systems, 563-564
target markets, 568
versioning systems, 564
Chronometer class, 155-156
circles, drawing, 219
classes
  AbsoluteLayout, 190
  AbstractAccountAuthenticator, 490
  AbstractThreadedSyncAdapter, 491
  AccountManager, 490, 497
  Activity, 71
  AdapterView
    ArrayAdapter class, 194-195
    binding data, 196
    CursorAdapter class, 195-196
event handling, 197
  AnalogClock, 156-157
  AnimationUtils, 227-228
AppWidgetProvider, 455
ArcShape, 220
ArrayAdapter, 194–195
AsyncTask, 292–293
audioRecorder, 348
AutoCompleteTextView, 139
BackupAgentHelper, 492
BasicGLThread, 372
BatteryManager, 419
Bitmap, 212
BitmapDrawable, 116
Bluetooth, 415
Bundle, 75
Button, 144
CacheManager, 307
Camera, 340
CameraSurfaceView, 337
Canvas, 207
Chronometer, 155–156
Configuration, 544
ConnectivityManager, 297
ConsoleMessage, 307
ContactsContract, 264
ContentValues, 268
ContentResolver, 276
Context class, 70
Activity instances, 71
application preferences, accessing, 70
application resources, retrieving, 70
reference website, 80
retrieving, 70
ContextMenu, 159–161
CookieManager, 307
CursorAdapter, 195–196
CustomGL2SurfaceView, 392
CustomRenderer, 392
DatePicker, 150–151
Dialog, 165
DigitalClock, 156
DisplayMetrics, 526
EditText, 138–142
auto-complete, 139–142
defining, 138
input filters, 142
long presses, 138–140
FileBackupHelper, 494–495
Gallery, 178
GalleryView, 194
GameAreaView
defining, 511
methods, 511, 513
multi-touch gestures, 516–519
single-touch gestures, 510–513
Geocoding, 319
address line queries, 320
named locations, 320–322
specific information queries, 320
GeomagneticField, 412
GeoPoint, 324
GestureDetector, 509
interfaces, 510
single-touch gesture support, 509–510
GestureOverlayView, 509
gestures, 509
GLDebugHelper, 373
GLES20, 392
GLSurfaceView
functionality, 388
implementing, 375–390
GPS satellite, 333
GPXService, 439
GridView, 194
Handler, 384
HorizontalScrollView, 201
HttpURLConnection, 289
InputMethodManager, 502
Intent, 78
ItemizedOverlay, 329-332, 333
Java, creating, 664
JavaScriptExtensions, 309
JNIEnv, 402
LayoutParams, 182
LinearGradient, 208
LinearInterpolator 230
ListView, 178, 194, 197-198
LocationListener, 316
LocationManager, 316
Log
importing, 59
methods, 59
MapController, 324
MarginLayoutParams, 182
Matrix, 213
MediaPlayer, 60
audio, playing, 348-349
methods, 61
video, 346
MediaRecorder
audio, 347-348
video, 343-345
MediaScannerConnection, 342
MediaStore content provider, 260
MotionEvent, 509
MultiAutoCompleteTextView, 141
NativeBasicsActivity.java, 400
NotificationManager, 425, 435
OnRatingBarChangeListener, 155
OrientationEventListener, 520
OvalShape, 219
Paint, 207
Parcelable, implementing, 446-449
Path, 220-222
PrefsListenerService, 458
ProgressBar class, 151-153
RadialGradient, 209
RatingBar class, 154-155
RecognizerIntent, 504-505
RectShape, 216
RemoteViews, 456-457
Renderer
functionality, 388
implementing, 375-390
RestoreObserver, 496-497
RingtoneManager, 351
RotateAnimation, 229
RoundRectShape, 217
ScaleAnimation, 229
ScaleGestureDetector
multi-touch gestures, 516
navigational gestures, 509
ScanResult, 413
ScrollView, 201
SearchManager, 470
SeekBar, 153-154
Sensor, 408
SensorEvent, 410
SensorManager, 408
Service, 439, 449
ServiceConnection, implementing,
445-446
shapes, 214
SharedPreferencesBackupHelper, 493
SimpleDataUpdateService, 458
SimpleOnGestureListener, 510
SimpleOrientationActivity, 520-521
SimpleViewDetailsActivity, 468
SlidingDrawer, 202-203
SmsManager
divideMessage() method, 362
default() method, 358
Spinner, 143-144
SQLite
databases, managing, 239
deleting, 249
SQLiteDatabase, 246-247
SQLiteOpenHelper, 250, 251-252
SQLiteQueryBuilder, 248-249
SweepGradient, 209
TabActivity, 198-200
TabHost, 178, 198
creating tabs from scratch, 200-201
TabActivity class implementation, 198-200
TelephonyManager, 354
TextView, 134-138
contextual links, creating, 136-138
height, 136
retrieving, 126
styles, applying, 169
text attribute, 135
width, 136
TimePicker, 151
TranslateAnimation, 230
Uri, 61
UriMatcher, 277-278
URL, 288
URLUtil, 307
View. See View class
ViewGroups, 178
attributes, 182
child View objects, adding, 178
layout classes, 178
subclass categories, 178
View container controls, 178
ViewSwitcher, 202
ViewTreeObserver
OnGlobalFocusChangeListener interface, 163
OnGlobalLayoutListener, 163
OnPreDrawListener interface, 163
OnTouchModeChangeListener interface, 162
ViewWithRedDot, 205-206
WallpaperManager, 342
WallpaperService, 462
WebBackForwardList, 307
WebChromeClient, 305-307
WebHistoryItem, 307
WebKit rendering engine, 307
WebSettings, 304
WebView. See also WebKit rendering engine
benefits, 307
deeper, adding, 305-307
content, loading, 302-304
event handling, 304-305
layouts, designing, 302
settings, configuring, 304
Web browsing, 301-302
WebViewClient, 304-305
WifiManager, 412-413
clear() method, 233
clearing logs, 654
click events, handling
AdapterView controls, 197
basic button controls, 146
image capturing, 340
long clicks, 164
menu option selection
context menus, 161
options menus, 159
client-server testing, 562
clock controls, 156-157
close() method, 250
closing SQLite databases, 250
cloud computing Wikipedia website, 497
Cloud to Device Messaging (C2DM), 438
code. See also development
diagnostics, 581
quality, 580
reviewing, 581
standards, 580-581
unit testing, 581-582
code obfuscation tools, 611
collapseColumns attribute, 191
colors, 111-112
# (hash symbol), 111
blinking light notifications, 430
formats, 111
paints, 207
antialiasing, 207
choosing, 207
gradients, 207-208
linear gradients, 208
Paint class, 207
radial gradients, 209
styles, 207
sweep gradients, 209
resource file example, 111
vertices (3D graphics), 377-378
<color> tag, 111
command-like gestures, 509
commit() method, 234
compare() method, 357
compatibility
alternative resources. See alternative resources
device differentiators, 523-524
forward, 554
hardware configuration support, 545-546
internationalization, 539-545
default language, configuring, 541-543
language alternative resources, 540-542
locales, 544-545
testing, 593
maximizing, 523-525
user interfaces, 525-531
Nine-Patch Stretchable images, 526-528
screen support, 526
working square principle, 528-531
versions, 546-548
API levels, finding, 546-547
backward compatibility with Java Reflection, 547-548
website, 549
completionThreshold attribute, 141
complex queries (SQL), 248-249
Configuration class, 544
configuring
alternative resources, 538
App Widget manifest file, 459
AVDs (Android Virtual Devices), 616-617
camera settings, 340-341
emulator locations, 62-63, 623-624
intent filters, 93
languages, 541-543
live wallpaper manifest file, 464-465
manifest files for market filtering, 599
multimedia optional features, 335-336
operating system for debugging, 30
OpenGL ES 2.0, 391
platform requirements, 90-92
device features, 91
input methods, 90
screen sizes, 91-92
creating from scratch, 200–201
TabActivity class implementation, 198–200
contains() method, 233
content providers
adding images to applications, 269
accessing images, 270–271
binding data to Gallery control, 272
data retrieval, 272
finding content with URIs, 271
gallery image retrieval, 273
retrieved images, viewing, 273–274
applications as, 274
data, adding, 278–279
data columns, defining, 276
deleting data, 280–281
interfaces, implementing, 275
manifest files, updating, 282
MIME types, returning, 281–282
queries, 276–277
updating, 279–280
URI pattern matching, 277–278
URIs, defining, 276
Browser, 259, 263
accessing, 263
querying for most visited
bookmarked sites, 263–264
built-in, 259
CallLog, 259, 261–263
access permissions, 262–263
tagging phone numbers with
custom labels, 262
Contacts, 259, 264
adding records, 267–268
deleting records, 269
private data, accessing, 264–266
querying, 266-267
updating records, 268-269
data
adding, 267-268
deleting, 269
retrieving, 272
updating, 268-269
interfaces, implementing, 275
live folder
enabling, 283
projections, 284
queries, handling, 482-484
URIs, defining, 283-284
MediaStore, 259, 260
classes, 260
data requests, 260-261
permissions, 95
registering, 94
Settings, 259, 267
UserDictionary, 259, 267
website, 285
content type handlers, 466-467
ContentResolver class, 276
ContentValues class, 268
Context class, 70
Activity instances, 71
application preferences, accessing, 70
application resources, retrieving, 70
reference website, 80
retrieving, 70
customizing
context menus, enabling, 159-161
ContextMenu class, 159-161
contextual links, creating, 136-138
controls, 134. See also classes
buttons, 144
basic, 144-146
check boxes, 144, 146-147
radio, 144, 148-149
toggles, 144, 147
clocks, 156-157
hardware for debugging, 30-31
layout, 134
OptionsMenu, 157-159
progress bars
Chronometer class, 155-156
RatingBar class, 154-155
SeekBar class, 153-154
progress indicators, 151-153
services, 443-444
source control systems, 563-564
choosing, 563-564
Eclipse IDE integration, 661
CookieManager class, 307
copy protection, 611
copying files
ADB commands, 650
DDMS File Explore, 641-642
/core files/directories, 52-51

source control systems, 563-564
copying files
ADB commands, 650
DDMS File Explore, 641-642
/core files/directories, 52-51

customizing
blinking light notifications, 431
dialogs, 168
fonts, 211-212
locales, 544
log filters, 663
notifications, 432
layouts, 433-434
text, 432-433
screen orientation example, 537-538
software keyboards, 502
CustomRenderer class, 392
CycleInterpolator, 230
Cygwin website, 398

D

dalvik
  Debug Monitor Server.  See DDMS
packages, 35
  virtual machine, 25
data synchronization, 491, 497
/data/app directory, 641
databases
devices
data storage, 556
devices, choosing, 556
functionality, 558
managing, 555-557
third-party, 558
persistent, creating, 250
SQLite.  See SQLite databases
student grade example, 675-682
  adding data to tables, 677
calculated columns, 680-682
deleting tables, 682
inging, 679
foreign keys, 678-679
multiple queries, 680
purpose, 675-676
querying, 677-678
schema, 676

  Students table, 676
tests table, 676
updating, 679
dataChanged() method, 496
/data/data/<package name>/cache/
directory, 641
/data/data/<package name>/databases
directory, 641
/data/data/<package name>/directory, 641
/data/data/<package name>/files/
directory, 641
/data/data/<package name>/
  shared_prefs/ directory, 641
date input retrieval, 150-151
date picker dialogs, 165
Datepicker class, 150-151
DDMS (Dalvik Debug Monitor Server), 36, 38
  application threads
    activity, monitoring, 638-639
    viewing, 637-638
availability, 635
debuggers, attaching, 638
Eclipse Perspective, 636
Emulator Control tab
  functionality, 642-643
  location fixes, 643
  SMS message simulation, 643
  voice calls, simulating, 643
features, 636-637
File Explorer
  browsing, 641
  copying files, 641-642
deleting files, 642
directory listing, 641
drag-and-drop support, 642
garbage collection, 639
heap activity, monitoring, 639-640
LogCat utility, 644
memory allocation, 640
processes, stopping, 640
screen captures, 645
stand-alone tool, 636
website, 38
debugging
ADB (Android Debug Bridge). See ADB
applications, 56–59
bugs
device specific, 582
reports, 569, 655–656
resolution process website, 32
configurations, creating, 53
on handsets, 65–66
hardware configuration, 30–31
operating system configuration, 30
registering applications as debuggable, 65
SDK, 32
user interfaces, 180
View object drawing issues, 180
DecelerateInterpolator, 230
default.properties file, 52
default resources, 132
defect tracking systems
defects
defining, 586–587
information, logging, 585–586
designing, 585
delete() method
contacts, 269
content provider data, 280–281
deleteFile() method, 235
deleting
Android Market applications, 609
content provider data, 269, 280–281
dialogs, 167
files, 235, 642
preferences, 233
SQLite database
records, 243–244
objects, 249
wallpapers, 343
deploying applications, 568. See also distributing applications
designing
Android Developers blog, 574
best practices
billing and revenue generation, 575
network diagnostics, 576–577
responsiveness, 573–574
stability, 573–574
third-party standards, 576
updates/upgrades ease, 577–578
user demands, 572
user interfaces, 572–573
websites, 584
extensibility, 565–566
handsets, 16
interoperability, 566–567
layouts, 125–127
locale support, 544–545
maintenance, 565–566
notifications, 434
rules, 571–572
security, 574
handling private data, 575
transmitting private data, 575
silly mistakes, avoiding, 578
tools, 578
destroying Activities, 75
developers
accounts
benefits, 609
creating, 26, 604–606
Distribution Agreement, 604
development

best practices, 579
code diagnostics, 581
code quality, 580
code reviews, 581
coding standards, 580-581
device specific bugs, 582
feasibility testing, 579-580
software processes, 579
unit testing, 581-582
websites, 584
common packages, 27
Eclipse IDE. See Eclipse IDE environment, testing, 43
adding projects to Eclipse workspace, 43-44
AVDs, creating, 44-46
launch configurations, creating, 46-48
running applications in Android emulator, 47-48
framework, 27-28
hardware configuration, 30-31
history, 17-18
mobile software
acquiring target devices, 560
applications, implementing, 567
architectures, 565
deployment, 568
device databases, 555-558
device limitations, 561, 564
extensibility, 565-566
interoperability, 566-567
iteration, 553, 570
maintenance, 565-566
overview, 551
project documentation, 562-563
project requirements, 553-554
quality assurance risks, 561-562
Rapid Application Development website, 570
source control systems, choosing, 563-564
support requirements, 568-569
target device identification, 558-560
testing, 567-568
third-party requirements, 555
use cases, 555
versioning systems, choosing, 564
waterfall approaches, 552, 570
Wikipedia website, 570
native versus third-party, 27
operating system configuration, 30
programming languages, 26
SDK upgrades, 31
silly mistakes, avoiding, 583
software requirements, 29
system requirements, 29
tools, 578, 583

devices

acquiring, 560
ADB commands, 648
battery monitoring, 417-420
Bluetooth
connections, 416-417
discovery, 416
paired devices, querying, 416
bugs, handling, 582
compatibility
alternative resources. See alternative resources
differentiators, 523-524
forward, 554
hardware configuration support, 545-546
internationalization, 539-545
maximizing, 523-525
user interfaces, 525-531
versions, 546-548
website, 549
connected, listing, 647-648
convergence, 13
custom binaries, installing, 659-660
databases
data storage, 556
devices, choosing, 556
functionality, 558
managing, 555-557
third-party, 558
debugging applications, 65-66
defect tracking systems, 585
defect information, logging, 585-586
defects, defining, 586-587
features, configuring, 91
files
browsing, 641
copying, 641-642, 650
deleting, 642
fragmentation, 587
hardware
accessing, 407
emulator support, 408
features, 408
identifying, 558-560
indicator lights, 430
languages, configuring, 541-543
limitations, 561, 564
locations, finding, 316-318
manufacturers, 16
customizations, 559
distribution partnerships, 611
market availability, 559-560
mobile operators, 17
Nexus One and Android Dev Phones website, 570
notifications support, 424
OpenGL ES compatibility, 368-369
preproduction, testing, 590
RAM size AVD hardware option, 619
real-life configurations, 588
screen captures, 645
sensors, 408
accelerometer, 410-411
accessing, 408, 409
availability, 409
calibrating, 410-411
data, reading, 409-410
most common, 408-409
orientations, 411-412
Sensor Simulator, 409
testing, 409
true north, finding, 412

**Dialog class, 165**

dialogs, 165

adding to activities, 166-167
alert, 165
character picker, 165
customizing, 168
date picker, 165
defining, 167
Dialog class, 165
dismissing, 167
initializing, 167
launching, 167
lifecycle, 166-167
progress, 165
removing, 167
time picker, 166
types, 165-166

**DigitalClock class, 156**

<dimen> tag, 112
dimensions, 112-113
   App Widgets, 454
canvases, 207
resource file example, 113
retrieving, 113
unit measurements, 112
directories, 235. See also files
   alternative resource qualifiers
      Android platform, 536
      applying, 532-537
bad examples, 536-537
case, 532
combining, 532
default resources, 536
dock mode, 534
good examples, 536
keyboard type and availability, 535
language and region code, 533
mobile country code, 533
mobile network code, 533
names, 532
navigation key availability, 535
navigation method, 536
night mode, 534
required strings, 533
screen aspect ratio, 534
screen orientation, 534
screen pixel density, 534
screen size, 533
text input method, 535
touch screen type, 535
cache files
   creating, 238-239
   retrieving, 236
core, 52-51
/data/app, 641
/data/data/<package name>, 641
/data/data/<package name>/cache/, 641
/data/data/<package name>/databases, 641
/data/data/<package name>/files/, 641
/data/data/<package name>/shared_prefs/, 641
files
   creating, 236, 238
   reading, 236
   reading byte-by-byte, 237
   retrieving, 236, 238
XML, 237
listing of, 641
/mnt/sdcard, 641
/mnt/sdcard/download/, 641
resources, 97-98
retrieving, 236
dismissDialog() method, 166, 167
dismissing dialogs, 167
display characteristics, finding, 526
**DisplayMetrics class, 526**
distributing applications, 568. See also publishing applications
   ad revenue, 612
   alternate marketplaces, 610-611
Android Market, 603-609
country requirements, 604
deleting applications, 609
developer accounts, 604-606, 609
Developer Distribution Agreement, 604
help, 607
licensing service, 604
publication, 608
refund policy, 608-609
sign-up website, 604
upgrading applications, 609
uploading applications, 606-608
website, 612
billing users, 611-612
generation methods, 575
testing, 594
considerations, 597-598
copy protection, 611
manufacturer/operator partnerships, 611
self-distribution, 609-610
divideMessage() method, 362
dock mode alternative resource qualifier, 534
documentation
  Javadoc-Style documentation, 667
  maintaining, 569
  porting, 563
  SDK, 33-34
  Searchable Configuration documentation website, 475
  software development, 562-563
    quality assurance plans, 562-563
    third-party, 563
  third-party, 563
  user interfaces, 563
doInBackGround() method, 292
doStartService() method, 441
downloading images from browsers, 271
DPad AVD hardware option, 619
Draw Nine-patch tool, 40, 527-528
<drawable> tag, 114
drawables, 113-114
drawBitmap() method, 213
drawFrame() method, 404
drawing
  3D graphics
    coloring vertices, 377-378
    cubes, 378
    lighting, 379-382
    texturing, 381-384
    vertices, 376-377
  android.graphics package, 230
  animations
    android.view.animation package, 226
    frame-by-frame, 223-225
    interpolators, 230
    loading, 227-228
    moving, 229-230
    rotating, 228-229
    scaling, 229
    transparency, 228
tweening. See tweening animations
types, 221-223
  bitmaps, 212, 213
    Bitmap class, 212
    scaling, 213
    transforming into matrixes, 213
canvases, 205-207
    Canvas object, 207
dimensions, 207
  paints, 207-210
    antialiasing, 207
colors, choosing, 207
gradients, 207-208
linear gradients, 208
Paint class, 207
radial gradients, 209
styles, 207
sweep gradients, 209
utilities, 210
red circle on black canvas example, 205-206
shapes
   arcs, 219-220
classes, 214
defining as XML resources, 214-215
defining programmatically, 215-216
ovals/circles, 219
paths, 220-222
round-corner rectangles, 217-218
squares/rectangles, 216-217
stars, 221-222
triangles on the screen, 375-376

Eclipse IDE, 30
  auto-complete, 664
  build errors, resolving, 667
download website, 29
Java code
   classes, creating, 664
   formatting, 664
   imports, organizing, 664-665
   methods, creating, 664
Javadoc-Style documentation, 667
layouts, designing, 125-127
log filters, creating, 663
manifest files, editing, 82
multiple file sections, viewing, 662
perspectives, 56, 662
Plug-In, 35
projects, adding, 43-44
refactoring code, 665
   Extract Local Variable tool, 666
   Extract Method tool, 666
   Rename tool, 665
reorganizing code, 667
resources, defining, 104-107
SimpleNDK application, 399-400
   exception handling, 402-403
   native code from Java, calling, 400-401
   parameters, handling, 401-402
   return values, handling, 401-402
source control services integration, 661
tabs
   repositioning, 661-662
unwanted, closing, 662
website, 41
windows
   maximizing, 662
   minimizing, 662
   open, limiting, 663
   side by side view, 662
edit() method, 233
editing
   manifest files, 82
      application-wide settings, 83-84
      Eclipse, 82
      manually, 84-86
      package-wide settings, 82-83
      permissions, 83
      test instrumentation, 83
strings, 107
EditText class, 138-142
   auto-complete, 139-142
defining, 138
input filters, 142
long presses, 138-140
EGL, initializing, 373-374
eglDestroyContext() method, 387
eglDestroySurface() method, 387
eglMakeCurrent() method, 387
eglTerminate() methods, 387
elapsedRealtime() method, 156
ellipsis (...), 136
ellipsize attribute, 136
emergency phone numbers, 357
ems attribute, 136
emulator, 37-38
  actual device testing, compared, 589-590
  AVDs (Android Virtual Devices), 615-616
    configuring, 616-617
    creating, 616-618
    hardware options, 618-620
    skin options, 618
  best practices, 613-614
  blinking lights, 430
  connected, listing, 647-648
console
  commands, 632
  connections, 628
  GPS coordinates, 630
  incoming call simulations, 628-629
  network status, monitoring, 631
  power settings, 631
  SMS message simulation, 629-630
  custom binaries, installing, 659-660
DDMS Emulator Control tab
  functionality, 642-643
  location fixes, 643
  SMS message simulation, 643
  voice calls, simulating, 643
debugging applications, 56-59
files
  browsing, 641
  copying, 641-642, 650
  deleting, 642
fun tips, 632
hardware support, 408
launching, 620-623
  running applications, 621-623
SDK and AVD Manager, 623
  startup options, 621
limitations, 590-591, 632-633
location, configuring,
  62-63, 318, 623-624
messaging between, 625-628
overview, 613
phone call simulation, 625
running applications through, 47-48,
  53-55
screen captures, 645
starting/stopping, 649-650
vibration, 429
website, 38
WiFi testing, 414
enhancing applications, 451-452
entire screen event handling, 162-163
  GlobalFocusChange, 163
  GlobalLayout, 163
  PreDraw, 163
event handling
  AdapterView controls, 197
  button clicks
    basic button controls, 146
    image capturing, 340
  entire screen events, 162-163
    GlobalFocusChange, 163
    GlobalLayout, 163
    PreDraw, 163
  focus changes, 164-165
  live wallpapers, 463
long clicks, 164
menu option selection
context menus, 161
options menus, 159
screen orientation changes, 520-521
touch mode changes, 161-162
WebView class, 304-305
execSQL() method, 241
Exerciser Monkey command-line tool, 594, 596
database data, 672
package files, 601-602
extending applications, 451-452
extensibility designs, 565-566
Extensible Markup Language. See XML
external Activities, launching, 77
external libraries, 92
Extract Local Variable tool, 666
Extract Method tool, 666
Extras property, 78
extreme programming website, 570

F

feasibility testing, 579-580
FileBackupHelper class, 494-495
fileList() method, 236
files. See also directories
application, backing up, 494-495
browsing, 641
cache
AVD hardware option, 620
creating, 238-239
retrieving, 236
copying
ADB commands, 650
DDMS File Explore, 641-642
core, 52-51
database formats, 669
deleting, 235, 642
image extensions, 114-115
listing, 236
locations, 50
manifest. See manifest files
opening, 235
packages, signing/exporting, 601-602
preferences, 231-234
raw, 121-122
resource, storing, 97
shared preferences, backing up, 493-494
SQL script files, creating, 673
storing, 101, 235
XML
App Widget definitions, 453-454
attributes, 120
in-application search files, 471
layouts, 120-121, 126, 173-175
live wallpaper definition, 464
parsing, 290-291
services permissions file, 443
shapes, defining, 214-215
SMS permissions, 358
telephony state information, 354
tweened animations, defining, 226
utility packages, 237
filter() method, 142
filters
input, 142
intents
configuring, 93
creating, 77
primary entry point activities, 92-93
registering, 469
remote interfaces, implementing, 446
logging, 652-653
market, 599, 612

finding
API levels, 546-547
audio, 350
content with URIs, 271
device locations, 316-318
display characteristics, 526
last known location, 63-64
multimedia, 350
preferences, 232
true north, 412

finish() method, 76

firmware upgrades, 569

first generation mobile phones, 9-10
first-to-market advantages, 23

Flash applications
Adobe AIR applications, building, 313
advantages/disadvantages, 311-312
enabling, 312-313

fling gestures, 515

focus changes, handling, 164-165

folders
assets, 52
gen, 52
live, 282
activities, 282, 481-482
components, 282-283
content provider queries, 482-484
creating, 481
enabling, 283
installing, 485-486
list with dates example, 285
manifest files, configuring, 484

overview, 480
picker information, 484
projections, 284
URIs, defining, 283-284
website, 487
res, 52
res/drawable-*/icon.png, 52
src, 52

fonts
chess font, 211
customizing, 211-212
default, 210
italic, 210
Monotype example, 210
Sans Serif example, 210
setFlags() method, 211
support, 210-211

forceError() method, 56

foreground attribute, 183

foregroundGravity attribute, 183

form layout example, 129-130

format strings, creating, 108

formatNumber() method, 357

formatting
colors, 111
database files, 669
images, 114-115
Java code, 664
phone numbers, 357-358
resource references, 122
strings, 107
video, 351

forward compatibility, 554

frame-by-frame animations, 116-117, 223-225

animation loops, naming, 224
genie juggling gifts example, 223-224
starting, 224
stopping, 224
FrameLayout views, 183-185
attributes, 183-185
XML resource file example, 184-185
framework
applications, 27-28
FAQ website, 449
SDK, 35
free market for applications, 22
functionality
applications, 97
Bluetooth, 414
DDMS (Dalvik Debug Monitor Server), 636-637
DDMS Emulator Control tab, 642-643
device databases, 558
GLSurfaceView class, 388
GPS, 316
manifest files, 81-82
OpenGL ES, 369
preferences, 232
Renderer class, 388
WebKit rendering engine, 308
G
galleries, 194
data-binding, 272
image retrieval, 273
Gallery class, 178
GalleryView class, 194
GameAreaView class
 defining, 511
 methods, 511, 513
 multi-touch gestures, 516-519
 single-touch gestures, 510-513
GC (Garbage Collector), 639
gen folder, 52
gen/com.androidbook.myfirstandroidapp/R.java file, 52
genie juggling gifts animation example, 223-224
Geocoder class, 319
 named locations, 320-322
 querying
 address lines, 320
 specific information, 320
gecoding, 318
 AVDs with Google APIs, 319
 GeoPoint objects, 324
 Location object names, retrieving, 319
 named locations, 320-322
 network connections, 321
 queries
 address lines, 320
 specific information, 320
GeomagneticField class, 412
GeoPoint objects, 324
GestureDetector class, 509
 interfaces, 510
 single-touch gesture support, 509-510
GestureListener interface
 methods, 515
 multi-touch implementation, 517
 single-touch implementation, 514
GestureOverlayView class, 509
gestures, 508-509
 android.gesture package, 509
classes, 509
 command-like, 509
 motion detection, 509
 multi-touch, 516-519
 ScaleGestureDetector class, 516
 SimpleMultiTouchGesture application example, 516-519
 natural, 518
navigational, 509
single-touch, 509-516
common, 509-510
detectors, 511
fling, 515
game screen example, 510-513
interpreting, 514
scroll, 515

getAddressLine() method, 320
getAll() method, 233
getAvailableLocales() method, 544
getBondedDevices() method, 416
getBoolean() method, 233
getCacheDir() method, 236
getCenter() method, 332
getConfiguredNetworks() method, 414
getContentResolver() method, 268
default() method, 358
defaultSensor() method, 409
deficientMinimumHeight() method, 343
deficientMinimumWidth() method, 343
dir() method, 236
displayMessageBody() method, 361
drawable() method, 116, 343
getExternalStoragePublicDirectory() method, 548
getFeatureName() method, 320
getFilesDir() method, 236
getFloat() method, 233
getAddress() method, 321
getInt() method, 233
getItem() method, 272
getItemId() method, 272
getLastNonConfigurationInstance() method, 539
getLocality() method, 320
getLocation() method, 63-64
getLong() method, 233
getMaxAddressLineIndex() method, 320
getMaxZoom() method, 341
getOrientation() method, 411-412
getResources() method, 70
getRoaming() method, 356
getSettings() method, 304
getSharedPreferences() method, 70
getString() method, 233
getSystemService() method
  ConnectivityManager, 297
  NotificationManager class, 425
  SensorManager class, 408
  TelephonyManager class, 354
  WifiManager class, 412-413
getTextBounds() method, 212
type() method, 281-282
getView() method, 272
getZoom() method, 341
getZoomRatios() method, 341
GIF (Graphics Interchange Format), 115
GL (Graphics Library), 374-375
glColorPointer() method, 377
glCompileShader() method, 394
GLDebugHelper class, 373
glDrawArrays() method, 376
glDrawElements() method, 376
GLES20 class, 392
glEsVersion attribute, 368
Global Positioning Services. See GPS
global searches, 478
GlobalFocusChange events, 163
GlobalLayout events, 163
GLSurfaceView class
  functionality, 388
  implementing, 375-390
gluLookAt() method, 375

 gluPerspective() method, 375

 gluUseProgram() method, 394

 GLUT (OpenGL Utility Toolkit), 375

 GNU

 Awk (Gawk) or Nawk website, 398

 General Public License Version 2 (GPLv2), 20

 Make 3.81 website, 398

 Google, 15

 Android Developer’s Guide, 41

 APIs Add-On, 35

 backup service, 492

 intents, 77

 Maps API key, 274, 325–326, 333

 maps integration

 AVDs with Google APIs, 62, 319

 emulator location, configuring, 62–63

 locations, mapping, 322–324

 GPLv2 (GNU General Public License Version 2), 20

 GPS (Global Positioning Services), 315–318

 application functionality, 316

 AVD hardware option, 619

 device locations, finding, 316–318

 emulator, locating, 318, 623–624

 satellite classes, 333

 GPXService class, 439

 gradients (paints), 207–208

 linear, 208

 radial, 209

 sweep, 209

 <grant-uri-permissions> tag, 95

 graphics. See images

 Graphics Interchange Format (GIF), 115

 gravity attribute

 LinearLayout views, 186

 RelativeLayout views, 187–189

 grids, 194

 GridView class, 194

 groups (permissions), 95

 GSM Modem AVD hardware option, 619

 Handler class, 384

 handling events. See event handling

 handsets. See devices

 hardware

 accessing, 407

 AVD configuration options, 618–620

 batteries, monitoring, 417–420

 Bluetooth

 available, finding, 415

 classes, 415

 connections, 416–417

 device discovery, 416

 enabling, 415–416

 functionality, 414

 implementation example, 417–418

 paired devices, querying, 416

 permissions, 415

 websites, 421

 configuration support, 545–546

 device sensors, 408

 accelerometer, 410–411

 accessing, 408, 409

 availability, 409

 calibrating, 410–411

 data, reading, 409–410

 most common, 408–409

 orientations, 411–412

 Sensor Simulator, 409

 testing, 409

 true north, finding, 412

 emulator support, 408
features, 408
WiFi
  access points, scanning, 412-413
  permissions, 412
  sensors, 412
  signal strength, 413
  testing, 414
hash symbol (#), 111
heap activity, monitoring, 639-640
hello-jni sample application, 399
Hierarchy Viewer, 39, 179-180
  drawing issues, debugging, 180
  launching, 179
  layout view, 180
  layouts, deconstructing, 180
  pixel perfect view, 180-181
  user interfaces, debugging, 180
hint attribute, 138
history of mobile software development
  applications, 17-18
  device convergence, 13
  first generation, 9-10
  first time waster games, 10
  Google, 15
  market, 14
  OHA (Open Handset Alliance)
    formation, 16
    manufacturers, 16
    mobile operators, 17
    website, 28
  OpenGL ES, 367
  proprietary platforms, 13
  WAP (Wireless Application Protocol), 11-13
horizontal progress bars, 152
horizontal scrolling, 201
HorizontalScrollView class, 201
hosts (App Widgets), 460
HTTP, 288
  asynchronous processing, 291-293
  images, viewing, 295-297
  latency, 298
  network calls with threads, 293-295
  network status, retrieving, 297
  reading Web data, 288-289
    errors, 289
    exception handling, 288
    permissions, 289
    URL class, 288
    URL queries, 289
    XML, parsing, 290-291
HttpURLConnection class, 289
HVGA skin, 618
hybrid project requirement methods, 554
IANA (Internet Assigned Numbers Authority), 467
icons (applications), 87, 599
IDEs (integrated development environments), 20
ImageButton, 146
images, 114-116
  3D. See 3D graphics
    accessing, 270-271
    adding, 115, 269
      accessing images, 270-271
      binding data to Gallery control, 272
      data retrieval, 272
      finding content with URIs, 271
      gallery image retrieval, 273
      retrieved images, viewing, 273-274
      android.graphics package, 230
animations, 116
android.view.animation package, 226
frame-by-frame, 116, 117, 223-225
helper utilities, 116
interpolators, 230
loading, 227-228
moving, 229-230
rotating, 228-229
scaling, 229
transparency, 228
tweening. See tweening animations
types, 221-223
BitmapDrawable class, 116
bitmaps, 212
Bitmap class, 212
drawing, 213
scaling, 213
transforming into matrixes, 213
buttons, 146
capturing with camera, 336-340
adding, 336-337
button click handler, 340
Camera object, instantiating, 337-338
camera parameters, 338
camera settings, 340-341
camera zoom controls, 341
CameraSurfaceView class, 337
layouts, 339
starting preview, 338-339
stopping preview, 338
takePicture() method, 339
downloading from browsers, 271
drawing
canvases, 205-207
paints, 207
red circle on black canvas example, 205-206
formats, 114-115
live wallpapers. See live wallpapers
NDK, 404
network, viewing, 295-297
Nine-Patch Stretchable
compatibility, 526-528
creating, 527-528
overview, 115
scaling, 527
OpenGL ES
3D graphics, 368
API documentation websites, 396
cleaning up, 387
device compatibility, 368-369
drawing on the screen, 375-376
EGL, initializing, 373-374
functionality, 369
GL, initializing, 374-375
GLDebugHelper class, 373
GLSurfaceView class, 388
history, 367
initializing, 369-370
Khronos OpenGL ES website, 396
main application thread communicating with OpenGL thread, 387
OpenGL thread talking to application thread, 386
overview, 367
Renderer class, 388
SurfaceView, creating, 370
thread, starting, 371-373
versions, 368
websites, 396
OpenGL ES 2.0, 391
configuring, 391
surface, requesting, 391-395
retrieved, viewing, 273-274
screen captures, 645
shapes
arcs, 219-220
classes, 214
defining as XML resources, 214-215
defining programmatically, 215-216
ovals/circles, 219
paths, 220-222
round-corner rectangles, 217-218
squares/rectangles, 216-217
stars, 221-222
sharing, 341-342
storing, 101
wallpapers
live wallpapers, 461-466
still images, 342-343
Images.Media class, 260
Images.Thumbnails class, 260
ImageUriAdapter, 272
IMEs (Input Method Editors), 499
importing
database data, 673-674
Log class, 59
in-application searches
activities, creating, 475-477
enabling, 470-472
manifest files, 477-478
Search buttons, 478
Searchable Configuration documentation website, 475
suggestions, 472-474
voice capabilities, 474-475
XML configuration file, 471
includeInGlobalSearch attribute, 478
#include tag, 124
indicator controls. See progress bars
initializing
dialogs, 167
EGL, 373-374
GL, 374-375
OpenGL ES, 369-370
shader programs, 392-394
static Activity data, 74
Text-To-Speech services, 507
input
date retrieval, 150-151
filters, 142
gestures, 508-509
android.gesture package, 509
classes, 509
command-like, 509
motion detection, 509
multi-touch, 516-519
natural, 518
navigational, 509
single-touch, 509-516
methods
configuring, 90
IMEs (Input Method Editors), 499
software keyboards, 499-502
technical articles website, 502
screen orientation changes, 520-522
text
alternative resource qualifier, 535
EditText controls, 138-142
prediction, 502
Spinner controls, 143-144
time retrieval, 151
trackballs, 519
Input Method Editors (IMEs), 499
InputMethodManager class, 502
inputType attribute, 501
insert() method
content provider data, 278-279
SQLite database records, 242
insertOrThrow() method, 242

installing
  App Widgets, 460-461
  applications, 651
  custom binaries, 659-660
  live folders, 485-486
  live wallpapers, 465-466
  NDK, 398

<integer> tag, 111
<integer-array> tag, 111

integer resources, 111
integrated development environments (IDEs), 20
Integrated Raster Imaging System Graphics Library (IRIS GL), 367
Integration of applications, 21-12
integration points, testing, 592-593

Intent class, 78
<intent-filter> tag, 92

intents
  activity transitions, 76
    creating, 77
    external Activities, launching, 77
    Google, 77
    new activities, launching, 76-77
    organizing with menus, 78
    passing additional information, 78
    Registry of Intents protocols, 77
  battery monitoring, 417
  Bluetooth, 415-416
  broadcasting, 79
filters
  configuring, 93
  creating, 77
  primary entry point activities, 92-93
  registering, 469
  remote interfaces, implementing, 446
  live folders, 481
  phone calls, making, 364
  processing with activities, 468
  receiving, 79
  reference website, 80
  SMS messages
    receiving, 360
    sending, 359
  speech recognition services, 506

interfaces
  content provider, implementing, 275
  ContentResolver, 276
  GestureDetector class, 510
    GestureListener
      methods, 515
      multi-touch implementation, 517
      single-touch implementation, 514
    OnChronometerTickListener, 156
    OnDoubleTapListener, 510
    OnFocusChangeListener, 164
    OnGestureListener, 510
    OnGlobalFocusChangeListener, 163
    OnGlobalLayoutListener, 163
    OnInitListener, 506
    OnLongClickListener, 164
    OnPreDrawListener, 163
    OnTouchModeChangeListener, 162
  remote, implementing, 444
    AIDL declaration, 444
    binder interface class name, 445
    code implementation, 445
    connecting/disconnecting services, 445-446
    disconnecting, 446
    intent filters, 446
    multiple interfaces, 445
    onBind() method, 445
    sharing across applications, 446
SensorEventListener, 409
SharedPreferences, 232, 233
SharedPreferences.Editor, 233–234
user. See user interfaces

internationalization, 539–545
default language, configuring, 541–543
language alternative resources, 540–542
device language and locale example, 541–542
dual language support example, 540–541
locales
    customizing, 544
    support, designing, 544–545
testing, 593

Internet access. See HTTP
Internet Assigned Numbers Authority (IANA), 467

interoperability, 566–567
interpolator attribute, 230
interpolators (animations), 230
IRIS GL (Integrated Raster Imaging System Graphics Library), 367
isAfterLast() method, 246
isDiscovering() method, 416
isEmergencyNumber() method, 357
isFinishing() method, 75
isSmoothZoomSupported() method, 341
Issue Tracker website, 32
isZoomSupported() method, 341
italic strings, 108
italic text, 210
ItemizedOverlay class, 329–333
iteration
    mobile software development, 553
query results, 246–247
Wikipedia website, 570

J

Java, 21
    code
        auto-complete, 664
        build errors, resolving, 667
        classes, creating, 664
        formatting, 664
        imports, organizing, 664–665
        methods, creating, 664
        obfuscation tools, 611
        refactoring, 665–666
        reorganizing, 667
Development Kit (JDK), 29
Javadoc-Style documentation, 667
JUnit, 35, 581–582
packages, 35
Platform website, 41
Reflection for backward compatibility, 547–548

Javadoc-Style documentation, 667
Java.net package, 299

JavaScript
    enabling, 304
    interface application, 308–312
        Button control click handler, 311
        JavaScript control, 311
        JavaScript namespace, 309
        JavaScriptExtensions class, 309
        onCreate() method, 309
        sample.html file JavaScript functions, 310–311
        web page, defining, 310
tutorial website, 314

JavaScriptExtensions class, 309
javax packages, 35
javax.xml package, 237
JDK (Java Development Kit), 29
JNIEnv object, 402
JPEG (Joint Photographic Experts Group), 115

K

keyboards
AVD hardware option, 619
software, 499-502
choosing, 500-502
customizing, 502
type and availability alternative resource qualifier, 535
Khronos Group, 367
Khronos OpenGL ES website, 396
killing Activities, 75

L

language support
alternative resources, 533
device language and locale example, 541-542
dual language example, 540-541
default, configuring, 541-543
locales
customizing, 544
support, designing, 544-545
websites, 549
last known location, finding, 63-64
launching
activities, 71, 76-77
ADB server processes, 648
configurations, creating, 46-48, 52-53
dialogs, 167
emulator, 620-623, 649-650
running applications, 621-623
SDK and AVD Manager, 623
startup options, 621
external activities, 77
files, 235
Hierarchy Viewer, 179
monkey tool, 656
services, 438
shell sessions, 649
layout_above attribute, 189
layout_alignBottom attribute, 188
layout_alignLeft attribute, 188
layout_alignParentBottom attribute, 188
layout_alignParentLeft attribute, 188
layout_alignParentRight attribute, 188
layout_alignParentTop attribute, 188
layout_alignRight attribute, 188
layout_alignTop attribute, 188
layout_below attribute, 189
layout_centerHorizontal attribute, 188
layout_centerInParent attribute, 187
layout_centerVertical attribute, 188
layout_column attribute, 191
layout_gravity attribute
FrameLayout views, 184
LinearLayout views, 186
layout_height attribute, 182
layout_margin attribute, 182
layout_span attribute, 191
layout_toLeftOf attribute, 189
layout_toRightOf attribute, 189
layout_weight attribute, 186
layout_width attribute, 182
LayoutParams class, 182
layouts, 123-124
alternative, 127
attributes, 181-182
built-in, 181
Button object margin example, 183
controls, 134
creating
  programmatically, 175-177
XML resources, 173-175
custom notifications, 433-434
deconstructing, 180
designing, 125-127
FrameLayout, 183-185
  attributes, 183-185
XML resource file example, 184-185
image capturing, 339
LinearLayout, 185-186
  attributes, 186
  examples, 182, 175-177
  horizontal orientation, 185
main.xml example, 123-124
multiple, 192
RelativeLayout, 186-190
  attributes, 187
  button controls example, 187
  views, 189
TableLayout, 190-192
  attributes, 191
  example, 190
  XML resource file example, 191-192
TextView object, retrieving, 126
ViewGroup subclasses, 178
Web, designing, 302
XML, accessing, 126
LBS (location-based services), 62-64, 315
  AVDs with Google APIs, creating, 62
  emulator location, configuring, 62-63, 623-624
  geocoding, 318
  address line queries, 320

AVDs with Google APIs, 319
GeoPoint objects, 324
Location object location names, retrieving, 319
named locations, 320-322
network connections, 321
specific information queries, 320
GPS, 315-318
  application functionality, 316
AVD hardware option, 619
device locations, finding, 316-318
emulator, locating, 318, 623-624
satellite classes, 333
ItemizedOverlay class, 333
last known location, finding, 63-64
locations, mapping
  application integration, 322-324
  Google Maps API Key, 325-326
  URIs, 322
maps
  panning, 326-327
  points of interest, marking, 327-332
  zooming, 327
permissions, 64
Proximity Alerts, 332
website, 333
Licensing Agreement, 32-33, 41
lifecycles
  activities, 72
    callbacks, 72-73
  destroying Activities, 75
initializing static activity data, 74
killing Activities, 75
releasing activity data, 74
retrieving activity data, 74
saving activity data, 74
saving state to Bundle objects, 75
stopping activity data, 74
dialogs, 166–167
services, 438, 449
lighting 3D graphics, 379-382
linear gradients, 208
LinearGradient class, 208
linear gradients, 208
LinearInterpolator class, 230
LinearLayout views, 134, 185-186
attributes, 186
example, 175-177, 182
horizontal orientation example, 185
lines attribute, 138
links (contextual), 136-138
Linux
Blog Man website, 649
Operating System, 23–24
lists, 194, 197-198
ListView class, 178, 194, 197-198
live folders, 282
activities, 282, 481-482
components, 282-283
content provider queries, 482-484
creating, 481
enabling, 283
installing, 485-486
list with dates example, 285
manifest files, configuring, 484
overview, 480
picker information, 484
projections, 284
URLs, defining, 283-284
website, 487
live server changes, managing, 569
live wallpapers, 461
application support, 462
creating, 462
examples, 461
installing, 465-466
manifest file, configuring, 464-465
service
creating, 462
implementing, 462
service engine implementation, 463
user events, handling, 463
website, 487
XML definition, 464
loadAndCompileShader() method, 393-394
loading animations, 227-228
locales
customizing, 544
ISO 3166-1-alpha-2 Regions website, 549
support, designing, 544-545
location-based services. See LBS
LocationListener class, 316
LocationManager class, 316
Log class
importing, 59
methods, 59
LogCat utility, 60, 644
clearing logs, 654
dates and times, 652
filters, 652–653
output redirection, 654
secondary logs, accessing, 654
viewing logs, 652
logo, 19
logs
clearing, 654
dates and times, 652
filters, 652–653, 663
LogCat utility, 60, 644
methods, 59
output redirection, 654
secondary, accessing, 654
support, adding, 59-60
viewing, 652
long click events, 164
low memory, killing Activities, 75
lowest common denominator method (project requirements), 553-554
low-risk porting, identifying, 569
LunarLander application, 40

M
Magic Eight Ball service, 438
magnetic fields, 412
maintenance designs, 565-566
main.xml layout example, 123-124
making phone calls, 362-364
managedQuery() method, 261
managing
Activity transitions with intents, 76
device databases, 555-558
live server changes, 569
manifest files, 81
activities
defining, 92
primary entry point, 92-93
App Widgets, configuring, 459
application settings
descriptions, 87
icons, 87
names, 87
versioning, 86
backup agents, registering, 495-496
broadcast receivers, registering, 93-94
content providers, registering, 94
editing, 82
application-wide settings, 83-84
Eclipse, 82
manually, 84-86
package-wide settings, 82-83
permissions, 83
test instrumentation, 83
external libraries, 92
functionality, 81-82
intent filters, configuring, 93
live folders, configuring, 484
live wallpapers, configuring, 464-465
MapView widget, 323
market filtering, configuring, 599
names, 81
permissions, registering
application-defined, 95
content providers, 95
required, 94-95
platform requirements, configuring, 90-92
device features, 91
input methods, 90
screen sizes, 91-92
searches, 477-478
services, registering, 93-94
settings, configuring, 96
system requirements, configuring, 87-90
updating, 282
website, 96
<manifest> tag, 89, 526
manufacturers, 16
device customizations, 559
distribution partnerships, 611
MapActivity, 324
MapController objects, 324
maps, 62-64
AVDs with Google APIs, creating, 62
eulator location, configuring, 62-63
ItemizedOverlay class, 333
last known location, finding, 63-64
locations, mapping
  application integration, 322-324
  Google Maps API Key, 325-326
  URIs, 322
panning, 326-327
points of interest, marking, 327-332
  ItemizedOverlay class, 329-332
  MapView widget, 327-329
Proximity Alerts, 332
zooming, 327
MapView widget, 323-324
  Google Maps API Key, 325-326
  manifest file, 323
  MapController objects, 324
  panning, 326-327
  permissions, 324
  points of interest, marking, 327-329
MarginLayoutParams class, 182
markers (maps), 327-332
  ItemizedOverlay class, 329-332
  MapView widget, 327-329
markets
alternatives, 610-611
Android Market
  country requirements, 604
  deleting applications, 609
  developer accounts, 604-606, 609
  Distribution Agreement, 604
  help, 607
  licensing service, 604
  publication, 608
  refund policy, 608-609
  signing-up website, 604
  upgrading applications, 609
  uploading applications, 606-608
  website, 612
device availability, 559-560
filters, 599, 612
first-to-market advantages, 23
packaging requirements, 599-600
target, choosing, 568
mascot, 19
Matrix class, 213
Max VM App Heap Size AVD hardware option, 620
maxEms attribute, 136
maximum SDK version, 90
maxLines attribute, 136
maxSdkVersion attribute, 88
measureAllChildren attribute, 184
measureText() method, 212
media. See multimedia
MediaController widget, 345-346
MediaPlayer class, 60
  audio, 348-349
  methods, 61
  video, 346
MediaRecorder class
  audio, 347-348
  video, 343-345
MediaScannerConnection class, 342
MediaStore content provider, 259, 260
classes, 260
data requests, 260-261
medium-size circular progress indicators, 152
memory allocation, monitoring, 640
menus
accessing, 120
activity organization, 78
context menus, enabling, 159-161
creating, 119
intent organization, 78
options menus, enabling, 157-159
resource file example, 119
storing, 101
XML attributes reference, 120

messaging (SMS)
3GPP Specifications website, 365
android.telephony package, 357
emulator messaging, 625-628
permissions, 358
receiving, 360-362
sending, 358-360
Wikipedia Write-Up website, 365

<meta-data> tag, 478

methods
addGlobalFocusChangeListener(), 163
addGlobalLayoutListener(), 163
addOnPreDrawListener(), 163
addOnTouchListener(), 162
addView(), 178
AppWidgetProvider class, 455-456
beginTransaction(), 244
bindService(), 438
boundCenterBottom(), 330
broadcastIntent(), 79
calculateAndDisplayFPS(), 385
cancel(), 428
cancelDiscovery(), 416
canDetectOrientation(), 521
close(), 250
compare(), 357
createBitmap(), 213
createScaledBitmap() method, 213
createTabContent(), 200
dataChanged(), 496
delete()
  contacts, 269
  content provider data, 280-281
deleteFile(), 235
dismissDialog(), 166, 167
divideMessage(), 166, 167
doInBackground(), 292
doStartService(), 441
drawBitmap(), 213
drawFrame(), 404
eglDestroyContext(), 387
eglDestroySurface(), 387
eglMakeCurrent(), 387
eglTerminate(), 387
elapsedRealtime(), 156
eexecSQL(), 241
file/directory management, 235-236
fileList(), 236
filter(), 142
finish(), 76
forceError(), 56
formatNumber(), 357
GameAreaView class, 511, 513
GestureListener interface, 515
getAddressLine(), 320
getAvailableLocales(), 544
getBondedDevices(), 416
getCacheDir(), 236
ggetCenter(), 332
ggetConfiguredNetworks(), 414
ggetContentResolver(), 268
ggetDefault(), 358
ggetDefaultSensor(), 409
ggetDesiredMinimumHeight(), 343
ggetDesiredMinimumWidth(), 343
ggetDir(), 236
ggetDisplayMessageBody(), 361
ggetDrawable(), 116, 343
ggetExternalStoragePublicDirectory(), 548
getFeatureName(), 320
getFilesDir(), 236
getFromLocationName(), 321
getItem(), 272
getItemId(), 272
getLastNonConfigurationInstance(), 539
getLocality(), 320
getLocation(), 63-64
getMaxAddressLineIndex(), 320
getMaxZoom(), 341
getOrientation(), 411-412
getResources(), 70
getRoaming(), 356
getSettings(), 304
getSharedPreferences(), 70
getSystemService()
   ConnectivityManager, 297
   NotificationManager class, 425
   SensorManager class, 408
   TelephonyManager class, 354
   WifiManager class, 412-413
getTextBounds(), 212
type(), 281-282
gView(), 272
gZoom(), 341
gZoomRatios(), 341
gColorPointer(), 377
gCompileShader(), 394
gDrawArrays(), 376
gDrawElements(), 376
gluLookAt(), 375
gluPerspective(), 375
gUseProgram(), 394
insert()
   content provider data, 278-279
   SQLite database records, 242
insertOrThrow(), 242
isAfterLast(), 246
isDiscovering(), 416
isEmergencyNumber(), 357
isFinishing(), 75
isSmoothZoomSupported(), 341
isZoomSupported(), 341
Java, creating, 664
loadAndCompileShader(), 393-394
logging, 59
managedQuery(), 261
measureText(), 212
MediaPlayer class, 61
moveToFirst(), 246
moveToNext(), 246
notify(), 425-426
onAccuracyChanged(), 409
onActivityResult(), 76
onAnimateMove()
   GameAreaView class, 513
   GestureListener interface, 515
onAnimateStep(), 513
onBind(), 445
onCheckedChangedListener(), 149
onClick(), 146
onConfigurationChanged(), 539
onContextItemSelected(), 161
onCreate(), 74
onCreateContextMenu(), 160
onCreateDialog(), 167
onCreateEngine(), 462
onCreateOptionsMenu(), 120, 158
onDateChanged(), 150
onDelete(), 456
onDestroy(), 75, 442-443
onDisabled(), 455
onDraw(), 205, 511
onDrawFrame(), 390, 404
onEnabled(), 455
onFling(), 515
onInit(), 507
onJsBeforeUnload(), 305
onKeyDown(), 386
onKeyUp(), 386
onListItemClick(), 197
onLongClick(), 164
onMove()

GameAreaView class, 513
GestureListener interface, 515
onOptionsItemSelected(), 159
onPageFinished(), 304
onPause(), 74
onPerformSync(), 491
onPostExecute(), 292
onPreExecute(), 292
onPrepareDialog(), 167
onRatingChanged(), 155
onResetLocation()

GameAreaView class, 513
GestureListener interface, 515
onResume(), 74
onRetainNonConfigurationInstance() 539
onSaveInstanceState(), 75
onScroll(), 515
onSensorChanged(), 409
onServiceConnected(), 445–446
onServiceDisconnected(), 445–446
onStart(), 440
onStartCommand(), 440
onStop(), 62
onTouchEvent(), 509, 511
onTouchModeChanged(), 162
onTrackballEvent(), 519
onUpdate(), 456, 458
openFileInput(), 235
openFileOutput(), 235, 236
openOrCreateDatabase(), 240
peekDrawable(), 343
playMusicFromWeb(), 61
populate(), 330
post(), 384
preferences, editing, 233–234
query()

applications as content providers, 276–277
SQLite databases, 246–247
rawQuery(), 249
readFromParcel(), 448
recordSpeech(), 505
registerForContextMenu(), 159
registerListener(), 409
remove()

preferences, 233
SQLite database records, 243
removeDialog(), 166, 167
requestRestore(), 496–497
requestRouteToHost(), 297
sendTextMessage(), 359
setAccuracy(), 317
setBase(), 155
setBuiltInZoomControls(), 304
setColor(), 207
setContentView(), 171
setCurrentTabByTag(), 201
setEGLContextClientVersion(), 392
setFilters(), 142
setFlags(), 211
setInterpolator(), 230
setJavaScriptEnabled(), 304
setLatestEventInfo(), 432
setLightTouchEnabled(), 304
setListAdapter(), 197
setOnClickListener(), 146
setOneShot(), 224
setOnFocusChangeListener(), 164
setOnLongClickListener(), 164
setOnTimeChangedListener(), 151
setParameters(), 340
setShader(), 207
setSupportZoom(), 304
setTheme(), 170
setTransactionSuccessful(), 244
setVideoURI(), 346
setWebChromeClient(), 305
setWebViewClient(), 304
setZoom(), 341
SharedPreferences interface, 233
showDialog(), 166, 167
speak(), 508
start(), 224
startActivity(), 76-77
startActivityForResult(), 76
startDiscovery(), 416
startScan(), 413
startService(), 438
startSmoothZoom(), 341
stop(), 224
stopScan(), 413
stopService(), 438
surfaceChanged(), 336
surfaceCreated(), 336, 371
takePicture(), 339
ThrowNew(), 402
toggleFPSDisplay(), 386
unbindService(), 446
update()
applications as content providers, 279-280
SQLite databases, 242
Uri parsing, 61
writeToParcel(), 448
MIME types
formats, 467
returning, 281-282
minEms attribute, 136
minimum SDK versions, 89
minLines attribute, 136
minSdkVersion attribute, 88
mnt/sdcard directory, 641
mnt/sdcard/download/ directory, 641
mobile country code alternative resource qualifier, 533
mobile network code alternative resource qualifier, 533
mobile operators, 17
mobile software development, 17-18
applications, implementing, 567
architectures, 565
deployment, 568
device databases
data storage, 556
devices, choosing, 556
functionality, 558
managing, 555-557
third-party, 558
device limitations, 561, 564
extensibility, 565-566
Google, 15
history
device convergence, 13
first generation, 9-10
first time waster games, 10
market, 14
categories, 335
finding, 350
formats website, 351
hardware, 335
images. See images
optional features, configuring, 335-336
support, adding, 60-62
video, 343
formats, 351
playing, 345-346
recording, 343-345
website, 351
multiple layouts, 192
multiple screens support website, 96
multiple themes, 170-171
multi-touch gestures, 516-519
ScaleGestureDetector class, 516
SimpleMultiTouchGesture application example, 516-519
MyFirstAndroidApp
AVD, creating, 51
build targets, 50
core files/directories, 52-51
debugging
eмуlator, 56-59
handset, 65-66
launch configurations, creating, 52-53
location-based services, adding, 62-64
AVDs with Google APIs, creating, 62
emulator location, configuring, 62-63
last known location, finding, 63-64
logging, adding, 59-60
MP3 playback support, adding, 60-62
names, 50
package name, 50
running in Android emulator, 53-55

N
named locations, 320-322
names
alternative resource directory qualifiers, 532
animation loops, 224
applications, 50, 87, 599
database fields, 251
manifest files, 81
packages, 50
permissions, 95
projects, 50
SDKs, 19
native applications versus third-party applications, 27
NativeBasicsActivity.java class, 400
natural gestures, 518
navigation. See also LBS
alternative resource qualifiers, 535
gestures, 509
NDK (Native Development Kit), 397
C/C++ advantages, 398
components, 398
disadvantages, 397-398
hello-jni sample application, 399
image performance, 404
installing, 398
platform support, 397
SimpleNDK application, 399-400
exception handling, 402-403
native code from Java, calling, 400-401
parameters, handling, 401-402
return values, handling, 401-402
websites, 405
network-driven applications, 565
networking
  asynchronous processing, 291-293
  calls with threads, 293-295
  cellular networks, emulating, 298
  diagnostics, 576-577
  fundamentals, 287
  HTTP, 288
    errors, 289
    permissions, 289
    reading data from the Web, 288-289
    URL queries, 289
  images, viewing, 295-297
  latency, 298
  status, retrieving, 297
WiFi
  access points, scanning, 412-413
  permissions, 412
  sensors, 412
  signal strength, 413
  testing, 414
XML, parsing, 290-291

Nexus One and Android Dev Phones website, 570

night mode alternative resource qualifier, 534

Nine-Patch Stretchable images
  compatibility, 526-528
  creating, 527-528
  overview, 115
  scaling, 527

NOAA: World Magnetic Model website, 421
nonprimitive storage, 257
NotePad application, 40
NotificationManager class, 425, 435
notifications
  audio, 431-432
  blinking lights, 430-431
    clearing, 431
    colors, 430
    customizing, 431
    precedence, 430
    testing, 430
    urgency, 430-431
  clearing, 428
  components, 424
  creating, 425
  customizing, 432
    layouts, 433-434
    text, 432-433
  designing, 434
  device support, 424
  examples, 423
  importance, 423
  NotificationManager class, 425
  reference websites, 435
  services, 442
  status bar, 424
    queues, 426-427
    text notification, creating, 425-426
    website, 435
  types, 423-424
  updating, 427-428
  vibration, 429
notify() method, 425-426

O

objects. See classes

OHA (Open Handset Alliance)
  formation, 16
  manufacturers, 16
  mobile operators, 17
  website, 28
onAccuracyChanged() method, 409
onActivityResult() method, 76
onAnimateMove() method
    GameAreaView class, 513
    GestureListener interface, 515
onAnimateStep() method, 513
onBind() method, 445
onCheckedChanged() method, 149
OnChronometerTickListener interface, 156
onClick() method, 146
onConfigurationChanged() method, 539
onContextItemSelected() method, 161
onCreate() method, 74
onCreateContextMenu() method, 160
onCreateDialog() method, 167
onCreateEngine() method, 462
onCreateOptionsMenu() method, 120, 158
onDateChanged() method, 150
onDeleted() method, 456
onDestroy() method, 75, 442-443
onDisabled() method, 455
onDoubleTap gesture, 510
onDoubleTapEvent gesture, 510
OnDoubleTapListener interface, 510
onDown gesture, 510
onDraw() method, 205, 511
onDrawFrame() method, 390, 404
onEnabled() method, 455
onFling gesture, 510
onFling() method, 515
OnFocusChangeListener interface, 164
OnGestureListener interface, 510
OnGlobalFocusChangeListener interface, 164
OnGlobalLayoutListener interface, 163
onInit() method, 507
OnInitListener interface, 506
onJsBeforeUnload() method, 305
onKeyDown() method, 386
onKeyUp() method, 386
onListItemClick() method, 197
onLongClick() method, 164
OnLongClickListener interface, 164
onMove() method
    GameAreaView class, 513
    GestureListener interface, 515
onOptionsItemSelected() method, 159
onPageFinished() method, 304
onPause() method, 74
onPerformSync() method, 491
onPostExecute() method, 292
OnPreDrawListener interface, 163
onPreExecute() method, 292
onPrepareDialog() method, 167
OnRatingBarChangeListener class, 155
onRatingChanged() method, 155
onResetLocation() method
    GameAreaView class, 513
    GestureListener interface, 515
onResume() method, 74
onRetainNonConfigurationInstance() method, 539
onSaveInstanceState() method, 75
onScroll gesture, 510
onScroll() method, 515
onSensorChanged() method, 409
onServiceConnected() method, 445-446
onServiceDisconnected() method, 445-446
onShowPress gesture, 510
onSingleTapConfirmed gesture, 510
onSingleTapUp gesture, 510
onStart() method, 440
onStartCommand() method, 440
onStop() method, 62
onTouchEvent() method, 509, 511
onTouchModeChanged() method, 162
OnTouchModeChangeListener interface, 162
onTrackballEvent() method, 519
onUpdate() method, 456, 458
Open Handset Alliance (OHA). See OHA
open source licensing, 18, 20
openFileInput() method, 235
openFileOutput() method, 235, 236
OpenGL ES
2.0, 391
    configuring, 391
    surface, requesting, 391-395
3D graphics. See 3D graphics
API documentation websites, 396
    cleaning up, 387
device compatibility, 368-369
drawing on the screen, 375-376
EGL, initializing, 373-374
functionality, 369
GL, initializing, 374-375
GLDebugHelper class, 373
GLSurfaceView class
    functionality, 388
    implementing, 375-390
history, 367
initializing, 369-370
Khronos OpenGL ES website, 396
overview, 367
Renderer class
    functionality, 388
    implementing, 375-390
SurfaceView, creating, 370
threads
    communication, 384-386
    starting, 371-373
    versions, 368
    website, 396
OpenGL Utility Toolkit (GLUT), 375
openOrCreateDatabase() method, 240
operating system configuration, 30
options menus, enabling, 157-159
OptionsMenu control, 157-159
org.apache.http packages, 35
org.json packages, 35
org.w3c.dom package, 35, 237
org.xmlpull package, 35, 237
org.xml.sax package, 35, 237
orientation attribute, 186
OrientationEventListener class, 520
orientation (screen)
    alternative resource qualifier, 534
    changes, 520-522
    customization example, 537-538
outsourcing testing, 596
ovals, drawing, 219
OvalShape object, 219
OvershootInterpolator, 230
packages
    android, 35, 131
    android.accounts, 489
    android.bluetooth, 415
    android.content, 232
    android.database.sqlite, 239
    android.gesture, 509
    android.graphics, 230
    android.graphics.drawable.shapes, 215
    android.hardware
        GeomagneticField class, 412
        SensorManager class, 408
    android.speech, 503
android.telephony, 354, 357
android.test, 582
android.view, 133
android.view.animation, 226
android.webkit, 307
android.widget, 134
dalvik, 35
java, 35
javax, 35
junit, 35
names, 50
org.apache.http, 35
org.json, 35
org.w3c.dom, 35
org.xmlpull, 35, 237
org.xml.sax, 35, 237
XML utility, 237

packaging applications

certification, 603
debugging, disabling, 600
exporting package files, 601-602
icons, 599
logging, disabling, 600
manifest files for market filtering, configuring, 599
market requirements, 599-600
names, 599
permissions, 600
release versions, testing, 603
signing package files, 600-602
target platforms, verifying, 599
versions, 599

Paint class, 207

paints, 207
antialiasing, 207
colors, choosing, 207
gradients, 207-208
linear, 208
radial, 209
sweep, 209
Paint class, 207
styles, 207
tools, 210

panning maps, 326-327

Parcelable classes, implementing, 446-449

parent views, 178

parsing XML, 290-291

Path class, 220-222

paths (shapes), 220-222
peekDrawable() method, 343

performance, testing, 594

permission attribute, 95
<permission> tag, 95

permissions

ad-hoc, 25
application defined, 25
audio recording, 348
battery monitoring, 417
Bluetooth, 415
Contacts private data, 264-266
content providers, 95, 262-263
groups, 95
location-based service, 64
manifest files, 83
MapView widget, 324
names, 95
networking, 289
packaging applications, 600
phone calls, making, 362
protection levels, 95
registering
application-defined, 95
required, 94-95
ringtones, 351
services, 443
SMS messages, 358
telephony, 354
vibration, 429
video recording, 345
wallpapers, 343
website, 96
WebView class, 301
WiFi, 412
persistent databases, creating, 250
field names, 251
SQLiteOpenHelper class, extending, 251-252
perspectives (Eclipse), 56, 662
PetListAdapter, 273
PetTracker application
binding data, 253-244, 254
field names, 251
SQLiteOpenHelper class, extending, 251-252
PetTracker2 application, 254-256
PetTracker3 application, images
accessing, 270-271
adding
  binding data to Gallery control, 272
data retrieval, 272
finding content with URIs, 271
Gallery image retrieval, 273
retrieved images, viewing, 273-274
phone calls
making, 362-364
receiving, 364-365
phone numbers
comparing, 357
emergency, 357
formatting, 357-358

PhoneGap project, 311
platforms
  alternative resource qualifiers, 536
  architecture, 23
  Linux Operating System, 23-24
  runtime environment, 25
  completeness, 18
  freedoms, 18
  market, 14
  mascot/logo, 19
  open source, 18, 20
  proprietary, 13-14
  requirements, configuring, 90-92
    device features, 91
    input methods, 90
    screen sizes, 91-92
  security, 25
    applications as operating system
    users, 25
    developer registration, 26
    permissions, 25
    trust relationships, 26
playing
  audio, 348-349, 620
  video, 345-346
playMusicFromWeb() method, 61
PNG (Portable Network Graphics), 114
points of interest, marking on maps, 327-332
  ItemizedOverlay class, 329-332
  MapView widget, 327-329
populate() method, 330
Portable Network Graphics (PNG), 114
porting
  documentation, 563
  low-risk projects, identifying, 569
post() method, 384
PreDraw events, 163
preferences
- accessing, 231–234
- adding, 232, 233–234
- applications, accessing, 70
data types, 231
deleting, 233
file formats, 234
finding, 232
functionality, 232
methods, 233
private, 232
reading, 232
shared, 232
updating, 234
PrefListenerService class, 458
preproduction devices, testing, 590
press-and-hold actions, 164
primary entry point activities, 92–93
private data
- handling, 575
- preferences, 232
processes
- ADB server, starting/stopping, 648
debuggers, attaching, 638
development, 579
reference website, 449
stopping, 640
programming languages, choosing, 26
progress bars
- adjusting, 153–157
  - exact values, viewing, 154
  - ratings, setting, 154
  - thumb selectors, 153–154
  - time passage, 155–156
clock controls, 156–157
horizontal, 152
medium-size circular, 152
placing in title bars, 153
ProgressBar class, 151–153
standard, 151
visibility, 153
progress dialogs, 165
ProgressBar class, 151–153
ProGuard website, 611
projections (live folders), 284
projects
- adding to Eclipse workspace, 43–44
- AVDs, creating, 51
- file locations, 50
- launch configurations, creating, 52–53
- names, 50
requirements
  - customization method, 554
  - hybrid approaches, 554
  - lowest common denominator method, 553–554
prompt attribute, 144
properties
  - Extras, 78
  - SQLite databases, configuring, 241
proprietary platforms, 13–14
protection levels (permissions), 95
<provider> tag, 94
providers
  - accounts, 490
  - App Widgets, 452, 455
  - content. See content providers
Proximity Alerts, 332
publishing applications. See also distributing applications
  - Android Market, 608
certification, 603
exporting package files, 601–602
packaging preparations
  - debugging, disabling, 600
icons, 599
logging, disabling, 600
manifest files for market filtering, configuring, 599
market requirements, 599-600
names, 599
permissions, 600
target platforms, verifying, 599
versions, 599
release versions, testing, 603
requirements, 598
signing package files, 600-602
\texttt{putBoolean()} method, 233
\texttt{putFloat()} method, 233
\texttt{putInt()} method, 233
\texttt{putLong()} method, 234
\texttt{putString()} method, 234

Q

quality assurance, 561-562. See also testing
code, 580
diagnostics, 581
reviews, 581
standards, 580-581
unit testing, 581-582
documentation, 562-563
third-party, 563
user interfaces, 563
killer apps, 594
\texttt{query()} method
applications as content providers, 276-277
SQLite databases, 246-247

querying
applications as content providers, 276-277
Bluetooth paired devices, 416

R

radial gradients, 209
RadialGradient class, 209
radio buttons, 144, 148-149
Rapid Application Development website, 570
rapid prototyping, 553
RatingBar class, 154-155
raw files, 121-122
raw queries (SQL), 249
\texttt{rawQuery()} method, 249
\texttt{readFromParcel()} method, 448

reading
device sensor data, 409-410
directory files, 236
byte-to-byte, 237
XML, 237
preferences, 232
Web data, 288-289
  errors, 289
  exception handling, 288
  permissions, 289
  URL class, 288
<li><code>&lt;receiver&gt;</code> tag, 94
receiving
  intents, 79
  phone calls, 364-365
  SMS messages, 360-362
RecognizerIntent class, 504-505
recording
  audio, 347-348, 619
  speech, 504-505
  video, 343-345
records (SQLite databases)
  deleting, 243-244
  inserting, 242
  updating, 242-243
recordSpeech() method, 505
rectangles, drawing, 216-217
rectangles with rounded corners, drawing, 217-218
RectShape object, 216
red circle on black canvas example, 205-206
refactoring code, 665
  Extract Local Variable tool, 666
  Extract Method tool, 666
referencing
  resources, 122-123
  system resources, 131
refunding applications (Android Market), 608-609
registerForContextMenu() method, 159
registering
  accounts, 490
  applications as debuggable, 65
backup agents, 495-496
backup services, 492
broadcast receivers, 93-94
content providers, 94
intent filters, 469
permissions
  application-defined, 95
  required, 94-95
services, 93-94
registerListener() method, 409
Registry of Intents protocols, 77
reinstalling applications, 651
RelativeLayout views, 186-190
  attributes, 187
  button controls example, 187
  XML resource file example, 189
release versions, testing, 603
releasing activity data, 74
remote backup services, 492
remote interfaces, implementing, 444
  AIDL declaration, 444
  binder interface class name, 445
  code implementation, 445
  connecting, 445-446
  disconnecting, 445-446
  intent filters, 446
  multiple interfaces, 445
  onBind() method, 445
  sharing across applications, 446
RemoteViews class, 456-457
remove() method
  preferences, 233
  SQLite database records, 243
removeDialog() method, 166, 167
Rename tool, 665
Renderer class
  functionality, 388
  implementing, 375-390
reorganizing code, 667
requestRestore() method, 496-497
requestRouteToHost() method, 297
res folder, 52
/res/drawable-*/ directory, 98
/res/drawable-*/*icon.png folders, 52
/res/layout/ directory, 98
res/layout/main.xml file, 52
resources
  accessing programmatically, 103
  adding, 98
  aliases, 123
  alternative, 102-103, 531
    configuration changes, handling, 539
    data retention, 539
    default application icon resources example, 531
    directory qualifiers, 532-537
    efficiency, 538-539
    hierarchy, 531
    internationalization, 540-542
    performance, 539
    programmatic configurations, 538
    screen orientation customization example, 537-538
    websites, 549
animations, 116
  android.view.animation package, 226
  frame-by-frame, 116, 117, 223-225
  helper utilities, 116
  interpolators, 230
  loading, 227-228
  moving, 229-230
  rotating, 228-229
  scaling, 229
  storing, 101
  transparency, 228
tweening, 116-118, 224-230
types, 221-223
applications, retrieving, 70
Boolean, 110
colors, 111-112
  # (hash symbol), 111
  formats, 111
resource file example, 111
default, 132
defined, 97
defining types with Eclipse, 104-107
dimensions, 112-113
  resource file example, 113
  retrieving, 113
  unit measurements, 112
directory hierarchy, 97-98
drawables, 113-114
images. See images
integer, 111
layout, 123-124
  alternative, 127
  attributes, 181-182
  built-in, 181
  Button object margin example, 183
  controls, 134
  creating programmatically, 175-177
  creating with XML resources, 173-175
custom notifications, 433-434
deconstructing, 180
designing, 125-127
FrameLayout, 183-185
image capturing, 339
LinearLayout, 185-186
main.xml example, 123-124
multiple, 192
RelativeLayout, 186-190
RelativeLayout views, 189
TableLayout, 190-192
TextView object, retrieving, 126
ViewGroup subclasses, 178
Web, designing, 302
XML, accessing, 126
menus, 119-120
accessing, 120
activity organization, 78
context menus, enabling, 159-161
creating, 119
intent organization, 78
options menus, enabling, 157-159
resource file example, 119
storing, 101
XML attributes reference, 120
raw files, 121-122
referencing, 122-123
selector, 116
storing, 97, 101
animations/graphics/files, 101
strings, 101
strings, 107
accessing, 108-109
arrays, 109-110
bold/italic/underlining, 108
editing, 107
formatting, 107, 108
resource file example, 108
storing, 101
styles, 127-130
applying with a parent, 169-170
attribute references example, 128
form layout example, 129-130
inheritance, 169-170
padding example, 168
paints, 207
previewing, 128
resource ids, 130
storing, 128
styles.xml example, 128
text size example, 168
TextView class, applying, 169
system
android package, 131
referencing, 131
themes, 131, 170-171
activities, 170
built-in, 171
entire screen, 170
multiple, 170-171
setTheme() method, 170
View objects, 170
types, 99-101
website, 132
XML files, 120-121
responsiveness, designing, 573-574
restore operations, 496-497
RestoreObserver class, 496-497
/res/values/ directory, 98
res/values/strings.xml file, 52
retrieving
Activity data, 74
application resources, 70
content provider data, 272
Context, 70
date input, 150-151
dimensions, 113
directories, 236
caches, 236
files, 236, 238
gallery images, 273
network status, 297
text input
   EditText controls, 138-142
   Spinner controls, 143-144
TextView object, 126
time input, 151

revenue
   ads, 612
   generation methods, 575
RingtoneManager object, 351
ringtones, 351

risks (software development)
device limitations, 561
quality assurance, 561-562
   client-server testing, 562
   early testing, 561
real-world testing limitations, 561-562
testing on the device, 561
target devices
   acquiring, 560
   identifying, 558-560
   manufacturer customizations, 559
market availability, 559-560

roaming state, 356
RotateAnimation class, 229

rotating animations, 228-229
round-corner rectangles, 217-218
RoundRectShape object, 217

Rubin, Andy, 16

Run configurations, creating, 52-53

runtime
   changes website, 549
   environment, 25

saving
   activity data, 74
   Activity state to Bundle objects, 75

ScaleAnimation class, 229
ScaleGestureDetector class
   multi-touch gestures, 516
   navigational gestures, 509

scaling
   animations, 229
   bitmaps, 213
   Nine-Patch Stretchable images, 527

ScanResult class, 413

screens
   aspect ratio alternative resource qualifier, 534
   compatibility. See user interfaces, compatibility
display characteristics, finding, 526
image captures, 645
multiple screen support websites, 96, 549
orientations
   alternative resource qualifiers, 534
   changes, 520-521, 522
   customization example, 537-538
pixel density alternative resource qualifiers, 534
sizes
   alternative resource qualifiers, 533
   configuring, 91-92
   screen aspect ratio alternative resource qualifier, 534

scroll gestures, 515
scrolling, 201

ScrollView class, 201

SD card AVD hardware option, 620
SDK (Software Development Kit)

accessibility framework, 502-503
android.speech package, 503
speech recognition services, 504-506
Text-To-Speech services, 503, 506-508
android.view package, 133
android.widget package, 134
availability, 20
buttons, 144
  basic, 144-146
  check boxes, 144, 146-147
  radio, 144, 148-149
  toggles, 144, 147
clock controls, 156-157
context menus, enabling, 159-161
data retrieval from users, 150-151
  EditText controls, 138-142
  Spinner controls, 143-144
dialogs, 165
  adding to activities, 166-167
  alert, 165
  character picker, 165
customizing, 168
date picker, 165
defining, 167
Dialog class, 165
dismissing, 167
initializing, 167
launching, 167
lifecycle, 166-167
progress, 165
removing, 167
time picker, 166
types, 165-166
documentation, 33-34
download website, 29, 41
emulator, launching, 623
framework, 35
Google APIs Add-On, 35
Hierarchy Viewer, 179-180
drawing issues, debugging, 180
launching, 179
layout view, 180
layouts, deconstructing, 180
pixel perfect view, 180-181
user interfaces, debugging, 180
License Agreement, 32-33, 41
names, 19
options menus, enabling, 157-159
pixel perfect view, 180-181
progress indicator controls
  Chronometer class, 155-156
  ProgressBar class, 151-153
  RatingBar class, 154-155
  SeekBar class, 153-154
services. See services
styles. See styles
times. See themes
time input retrieval, 151
tools, 35-36
  ADB. See ADB
  ADT plug-in, 35-36
  AVD Manager, 36-37
  DDMS, 36 See also DDMS
  Draw Nine-patch, 40
  Eclipse Plug-In, 35
eulator. See emulator
Hierarchy Viewer. See Hierarchy Viewer
troubleshooting, 32
updating, 23, 31
user interface controls, 134
  layout, 134
  TextView, 134-138
versions, 87-90
  maximum, 90
  minimum, 89
  target, choosing, 89
  View class, 133

Search buttons, 478

Searchable Configuration documentation website, 475

searches, 469-470
  global, 478
  in-application, 470-471
    activities, creating, 475-477
    enabling, 471-472
    manifest files, 477-478
  Search buttons, 478
  Searchable Configuration documentation website, 475
  suggestions, 472-474
  voice capabilities, 474-475
  XML configuration file, 471
  website, 487

SearchManager class, 470

searchSuggestAuthority attribute, 472

searchSuggestIntentAction attribute, 472

searchSuggestIntentData attribute, 472

searchSuggestPath, 472

searchSuggestSelection attribute, 472

searchSuggestThreshold attribute, 472

Secure Sockets Layer (SSL), 288

security, 25, 574
  applications as operating system users, 25
  certifying applications, 603
  copy protection, 611
  developer registration, 26
  permissions
    ad-hoc, 25
    application defined, 25, 95
  audio recording, 348
  battery monitoring, 417
  Bluetooth, 415
  CallLog content, 262-263
  Contacts private data, 264-266
  content providers, 95
  groups, 95
  location-based service, 64
  manifest files, 83
  MapView widget, 324
  names, 95
  networking, 289
  packaging applications, 600
  phone calls, making, 362
  protection levels, 95
  required, registering, 94-95
  ringtones, 351
  services, 443
  SMS, 358
  telephony, 354
  vibrations, 429
  video recording, 345
  wallpapers, 343
  website, 96
  WebView class, 301

WiFi, 412

private data
  handling, 575
  transmitting, 575

signing package files, 600-602

trust relationships, 26

website, 96

SeekBar class, 153-154
  exact values, viewing, 154
  simple thumb selector example, 153-154

selector resources, 116
self-distribution, 609-610
sending SMS messages, 358-360
sendTextMessage() method, 359
Sensor class, 408
Sensor Simulator, 409
SensorEvent class, 410
SensorEventListener interface, 409
SensorManager class, 408
sensors
device
  accelerometer, 410-411
  accessing, 408, 409
  availability, 409
  calibrating, 410-411
  data, reading, 409-410
  most common, 408-409
  orientations, 411-412
  Sensor Simulator, 409
testing, 409
  true north, finding, 412
website, 421
WiFi, 412
servers
  ADB, starting/stopping, 648
  application distribution, 609-610
testing, 591-592
Service class, 439, 449
ServiceConnection object, implementing, 445-446
<service> tag
  broadcast receivers, registering, 94
  WallpaperService class, 464
services, 437
  Android Market licensing, 604
  App Widget update, creating, 458-459
  AudioManager, 349
  backup, 491
application files, 494-495
archived data, wiping, 655
backup agent implementations, 492-493
controlling with ADB, 654-655
forcing restores, 655
registering backup agents, 495-496
remote, choosing, 492
requesting backups, 496
restore operations, 496-497
scheduling, 655
shared preferences files, backing up, 492-493
C2DM, 438
communicating data to users
  notifications, 442
  toast messages, 442
connections, 438
controlling, 443-444
creating, 439-443
  background processing, 441
  doStartService() method, 441
  GPXSService class implementation, 440
  onStart()/onStartCommand() methods, 440
Service class, defining, 439
criteria, 437
examples, 79
GPS, 315-318
  application functionality, 316
  AVD hardware option, 619
device locations, finding, 316-318
  emulator, locating, 318, 623-624
  satellite classes, 333
implementing, 438
LBS (location-based services). See LBS
lifecycle, 438, 449
live wallpapers, 461
creating, 462
implementing, 462
Magic Eight Ball, 438
overview, 79
Parcelable classes, implementing, 446-449
registering, 93-94
remote interfaces, implementing, 444
AIDL declaration, 444
binder interface class name, 445
code implementation, 445
connecting/disconnecting, 445-446
disconnecting, 446
intent filters, 446
multiple interfaces, 445
onBind() method, 445
sharing across applications, 446
Service class website, 449
SimpleDroidWallpaper, 462
speech recognition, 504-506
starting, 438
stopping, 438, 442-443
telephony
information, retrieving, 356
state, 355-356
testing, 591-592
Text-To-Speech, 503, 506-508
converting text into sound files, 508
initializing, 507
language settings, 507
OnInitListener interface, 506
updating, 442
XML permissions file, 443
setAccuracy() method, 317
setBase() method, 155
setBuiltInZoomControls() method, 304
setColor() method, 207
setContentView() method, 171
setCurrentTabByTag() method, 201
setEGLContextClientVersion() method, 392
setFilters() method, 142
setFlags() method, 211
setInterpolator() method, 230
setJavaScriptEnabled() method, 304
setLatestEventInfo() method, 432
setLightTouchEnabled() method, 304
setListAdapter() method, 197
setOnClickListner() method, 146
setOneShot() method, 224
setOnFocusChangeListener() method, 164
setOnLongClickListener() method, 164
setOnTimeChangedListener() method, 151
setParameters() method, 340
setShader() method, 207
setSupportZoom() method, 304
setTheme() method, 170
Settings content provider, 259, 267
setTransactionSuccessful() method, 244
setVideoURI() method, 346
setWebChromeClient() method, 304
setZoom() method, 341
shader programs, initializing, 392-394
ShapeDrawable class, 214
shapes
arcs, 219-220
classes, 214
defining
programmatically, 215-216
XML resources, 214-215
ovals/circles, 219
paths, 220-222
round-corner rectangles, 217-218
squares/rectangles, 216-217
stars, 221-222
SharedPreferencesBackupHelper class, 493
SharedPreferences.Editor interface, 233-234
shared preferences files, backing up, 493-494
SharedPreferences interface, 232, 233
sharing
  audio, 349-350
  images, 341-342
  preferences, 232
  remote interfaces, 446
shell commands, 649-650
backup services
  archived data, wiping, 655
  forcing restores, 655
  scheduling, 655
bug reports, 655-656
custom binaries, installing, 659-660
emulator, starting/stopping, 649-650
issuing single, 649
listing, 660
monkey tool
  event types, weighting, 657-658
  launching, 656
  listening, 656-657
  seed feature, 658
  throttle, 658
shell sessions, starting, 649
Short Message Service. See SMS messages
showDialog() method, 166, 167
shrinkColumns attribute, 191
signals
  loss, testing, 589
  strength, monitoring, 356-357
signing package files, 600-602
silly mistakes, avoiding
  designs, 578
  development, 583
  testing, 595
SimpleDatabase application
  file, accessing, 240
  openOrCreateDatabase() method, 240
  properties, configuring, 241
SimpleDataUpdateService class, 458
SimpleDroidWallpaper service, 462
SimpleMultiTouchGesture application example, 516-519
SimpleNDK application
  exception handling, 402-403
  native code from Java, calling, 400-401
  parameters, handling, 401-402
  return values, handling, 401-402
SimpleOnGestureListener class, 510
SimpleOrientationActivity class, 520-521
SimpleSearchableActivity, 475-477
SimpleViewDetailsActivity class, 468
single-touch gestures, 509-516
  common, 509-510
  detectors, 511
  fling, 515
  game screen example, 510-513
  interpreting, 514
  scroll, 515
sizing
  App Widgets, 454
  screens
    alternative resource qualifiers, 533
    configuring, 85-92
    screen aspect ratio alternative resource qualifier, 534
text, 136, 212
  wallpapers, 343
skins (AVDs), 618
sliding drawers, 202-203
SlidingDrawer class, 202-203
SMS (Short Message Service) messages, 357
   3GPP Specifications website, 365
   android.telephony package, 357
   emulator messaging, 625-628
   permissions, 358
   receiving, 360-362
   sending, 358-360
   Wikipedia Write-Up website, 365
SmsManager class
   divideMessage(), 362
   getDefault() method, 358
Snake application, 40
   adding to Eclipse workspace, 43-44
   AVD, creating, 44-46
   launch configurations, creating, 46-48
   running in Android emulator, 47-48
software
   development. See mobile software
development
   integration, testing, 588-589
   keyboards, 499-502
      choosing, 500-502
      customizing, 502
   requirements, 29
source control systems
   choosing, 563-564
   Eclipse IDE integration, 661
speak() method, 508
speech recognition services, 504-506
Spinner class, 143-144
SQL
   commands, executing, 674
   script files, creating, 673
   SQLzoo.net website, 258
sqlite3 command-line tool, 240, 656, 670
   ADB shell, launching, 670
   commands, listing, 675
   data
      dumping, 672-673
      exporting, 672
      importing, 673-674
   limitations, 675
   SQL commands, executing, 674
   SQL script files, creating, 673
SQLite databases
   connecting/disconnecting, 670-671
      schemas, 672
   tables
      indices, 671
      listing, 671
      schemas, 672
SQLiteDatabase class, 246-247
SQLite databases, 239
   binding data to user interfaces,
      253-244
      adapter with ListView, 254-256
      adapters, 254
   closing, 250
   creating, 240
      file, accessing, 240
      openOrCreateDatabase() method, 240
      properties, configuring, 241
   data
      dumping, 672-673
      exporting, 672
      importing, 673-674
      SQL script files, creating, 673
   deleting, 250
   file formats, 669
   limitations, 675
   listing available, 671
management classes, 239
nonprimitive storage, 257
persistent, creating, 250
field names, 251
SQLiteOpenHelper class, extending, 251-252
querying, 244
complex queries, 248-249
cursors, 245
filtering results, 248
iterating results, 246-247
query() method, 246-247
raw queries, 249
WHERE clauses, 247
records
deleting, 243-244
inserting, 242
updating, 242-243
schemas, 672
sqlite3 command-line tool, 240, 656, 670
ADB shell, launching, 670
command listing, 675
dumping data, 672-673
exporting data, 672
importing data, 673-674
limitations, 675
SQL commands, executing, 674
SQL script files, creating, 673
SQLite databases, connecting/disconnecting, 670-671
SQLite databases, schemas, 672
tables, 671-672
storing, 669
student grade example, 675-682
adding data to tables, 677
calculated columns, 680-682
deleting tables, 682
editing, 679
foreign keys, 678-679
multiple queries, 680
purpose, 675-676
querying, 677-678
schema, 676
Students table, 676
Tests table, 676
tables
creating, 241-242
deleting, 249
indices, 671
listing available, 671
schemas, 672
transactions, 244
triggers, 241-242
SQLiteOpenHelper class, 250-252
SQLiteQueryBuilder class, 248-249
SQLite website, 258
SQLZoo.net website, 258
squares, drawing, 216-217
src folder, 52
src/com.androidbook.myfirstandroidapp/MyFirstAndroidAppActivity.java file, 52
SSL (Secure Sockets Layer), 288
stability, designing, 573-574
stand-alone applications, 565
standard progress bars, 151
stars, drawing, 221-222
start() method, 224
startActivity() method, 76-77
startActivityForResult() method, 76
startDiscovery() method, 416
starting. See launching
startScan() method, 413
startService() method, 438
startSmoothZoom() method, 341
status bar notifications, 424
  clearing, 428
  queues, 426-427
  text notification, creating, 425-426
  updating, 427-428
  website, 435
stop() method, 224
stopping
  activity data, 74
  ADB server processes, 648
  animations, 224
  camera preview, 338
  emulator, 649-650
  processes, 640
  services, 438, 442-443
  shell sessions, 649
stopScan() method, 413
stopService() method, 438
storing
  databases, 669
  files, 235
  nonprimitive types in databases, 257
  resources, 97, 101
    animations/graphics/files, 101
    strings, 101
  styles, 128
stress testing applications
  events
    listening, 656-657
    types, weighting, 657-658
  monkey tool, 656
  repeating events, 658
  throttle, 658
stretchColumns attribute, 191
<string> tag, 107
<string-array> tag, 109
strings, 107
accessing, 108-109
arrays, 109-110
bold/italic/underlining, 108
editing, 107
formatting, 107, 108
resource file example, 108
storing, 101
student grades database, 675-682
  calculated columns, 680-682
  editing, 679
  foreign keys, 678-679
  multiple queries, 680
  purpose, 675-676
  querying, 677-678
  schema, 676
  tables
    data, adding, 677
    deleting, 682
    Students table, 676
    Tests, 676
    updating, 679
<style> tag, 128
styles, 127-130, 168-170
  applying with a parent, 169-170
  attribute references example, 128
  form layout example, 129-130
  inheritance, 169-170
  padding example, 168
  paints, 207
  previewing, 128
  resource ids, 130
  storing, 128
  styles.xml example, 128
  text size example, 168
  TextView class, applying, 169
support requirements, 568
  documentation, 569
  firmware upgrades, 569
  live server changes, 569
  low-risk porting, identifying, 569
  user crash/bug reports, 569
<supports-screen> tag, 91, 526
surfaceChanged() method, 336
surfaceCreated() method, 336, 371
SurfaceView widget, 370-371
sweep gradients, 209
SweepGradient class, 209
switchers, 202
Sync Adapter example application, 491, 497
class, creating, 198
<activity>, 92
<appwidget-provider>, 454
<bool>, 110
<color>, 111
<dimen>, 112
<drawable>, 114
<grant-uri-permissions>, 95
<include>, 124
<integer>, 111
<integer-array>, 111
<intent-filter>, 92
<manifest>, 89
<meta-data>, 478
<permission>, 95
<provider>, 94
<receiver>, 94
<service>
  broadcast receivers, registering, 94
  WallpaperService class, 464
<string>, 107
<string-array>, 109
<style>, 128
<supports-screen>, 91, 526
<uses-configuration>
  input methods, 90
  trackballs, 519
<uses-feature>
  device features, configuring, 91
  GPS, 316
<uses-permission>, 94
<uses-sdk>, 88
<wallpaper>, 464
Manifest, 526
takePicture() method, 339

target devices
acquiring, 560
identifying, 558-560
manufacturer customizations, 559
market availability, 559-560

targetSdkVersion attribute, 88, 89

telephony
call states
listening for changes, 355
permissions, 354
querying, 354-355
roaming, 356
connection speeds, monitoring, 356-357
permissions, 354
phone calls
making, 362-364
receiving, 364-365
phone numbers
comparing, 357
emergency, 357
formatting, 357-358
services
information, retrieving, 356
state, 355-356
signal strength, monitoring, 356-357
SMS messages, 357
3GPP Specifications website, 365
android.telephony package, 357
permissions, 358
receiving, 360-362
sending, 358-360
Wikipedia Write-Up website, 365
TelephonyManager class, 354

TelephonyManager class, 354

testing. See also quality assurance
applications, 567-568
automating, 590
backup services, 594
best practices, 585
billing, 594
black box, 591
build acceptance tests, 589
client-server, 562
conformance, 593
coverage, maximizing, 589
defect tracking systems, 585
defect information, logging, 585-586
defects, defining, 586-587
development environment, 43
adding projects to Eclipse workspace, 43-44
AVDs, creating, 44-46
launch configurations, creating, 46-48
running applications in Android emulator, 47-48
devices, 561
eyeal testing, 561
fragmentation, 587
sensors, 409
emulator. See emulator environments, 587
feasibility testing, 579-580
installations, 593
integration points, 592-593
internationalization, 593
outsourcing, 596
performance, 594
preproduction devices, 590
priorities, 588
quality, 594
real-life
devices, 588
limitations, 561-562
reference websites, 596
release versions, 603
servers, 591-592
services, 591-592
signal loss, 589
silly mistakes, avoiding, 595
software integration, 588-589
specialized scenarios, 592
starting states, 588
stress testing applications
  event listening, 656-657
  event types, weighting, 657-658
  monkey tool, launching, 656
  repeating events, 658
  throttle, 658
third-party
  firmware upgrades, 587
  standards, 592
tools, 595
unexpected events, 594
unit testing, 581-582
upgrades, 593
usability, 592
vibration, 429
white box, 591
WiFi, 414

fonts
  chess font, 211
  customizing, 211-212
  default, 210
  Monotype example, 210
Sans Serif example, 210
setFlags() method, 211
support, 210-211
input methods
  alternative resource qualifier, 535
  IMEs (Input Method Editors), 499
  software keyboards, 499-502
  text prediction, 502
input retrieval
  EditText controls, 138-142
  Spinner controls, 143-144
italic, 210
notifications, customizing, 432-433
prediction, 502
sizing, 136, 212
status bar notification, creating, 425-426

text attribute, 135
textOn/textOff attributes, 147
Text-To-Speech services, 503, 506-508
  converting text into sound files, 508
  initializing, 507
  language settings, 507
  OnInitListener interface, 506

texting 3D graphics, 381-384

TextView class, 134-138
  contextual links, creating, 136-138
  height, 136
  retrieving, 126
  styles, applying, 169
  text attribute, 135
  width, 136

themes, 131, 170-171
  applying
    activities, 170
    entire screen, 170
    View objects, 170
  built-in, 171
multiple, 170-171
setTheme() method, 170

**third-party**
applications versus native applications, 27
design standards, 576
device databases, 558
documentation, 563
firmware considerations, 587
software development requirements, 555
testing standards, 592

**threads**
application
activity, monitoring, 638-639
viewing, 637-638
OpenGL
starting, 371-373
talking to application thread, 384-385
reference website, 449
**ThrowNew() method, 402**

time
input retrieval, 151
passage progress bars, 155-156
picker dialogs, 166
**TimePicker class, 151**
**Toast messages, 146, 442**
**toggle buttons, 144**
**toggleFPSDisplay() method, 386**
**toggles, 147**
**tools, 35-36**
ADB. See ADB
ADT plug-in, 35-36
animation helper, 116
Asset Packaging, 98
AVD Manager, 36-37
code obfuscation, 611
DDMS. See DDMS
design, 578
development, 583
Draw Nine-patch, 40, 527-528
Eclipse Plug-In, 35
emulator. See emulator
Exerciser Monkey, 594, 596
Extract Local Variable, 666
Extract Method, 666
GLUT (OpenGL Utility Toolkit), 375
Hierarchy Viewer, 39, 179-180
drawing issues, debugging, 180
launching, 179
layout view, 180
layouts, deconstructing, 180
pixel perfect view, 180-181
user interfaces, debugging, 180
LogCat, 60, 644
clearing logs, 654
dates and times, 652
filters, 652-653
output redirection, 654
secondary logs, accessing, 654
viewing logs, 652
monkey
event types, weighting, 657-658
launching, 656
listening, 656-657
seed feature, 658
throttle, 658
website, 659
paints, 210
Rename, 665
sqlite3, 240, 670, 656
ADB shell, launching, 670
command listing, 675
database connections, 670-671
database schemas, 672
dumping data, 672-673
exporting data, 672
importing data, 673-674
limitations, 675
listing available databases, 671
SQL commands, executing, 674
SQL script files, creating, 673
table indices, 671
table schemas, 672
tables, listing, 671
testing, 595
XML packages, 237
touch gestures. See gestures
touch screen
alternative resource qualifier, 535
AVD hardware option, 619
mode changes, 161-162
trackballs, 519, 619
transactions (SQL), 244
transformations (tweened animations)
alpha transparency, 228
defining, 224
interpolators, 230
moving, 229-230
rotating, 228-229
scaling, 229
transitions (activities), 76
external Activities, launching, 77
intent action/data types, 77
new activities, launching, 76-77
passing additional information, 78
TranslateAnimation class, 230
transmitting private data, 575
transparency (animations), 228
triangles, drawing, 375-376
triggers (SQL), 241-242
troubleshooting
backup services, 497
build errors, 667
device specific bugs, 582
SDK, 32
signal loss, 589
support requirements, 568
documentation, 569
firmware upgrades, 569
live server changes, 569
low-risk porting, identifying, 569
user crash/bug reports, 569
true north, finding, 412
trust relationships, 26
tweening animations,
116, 117-118, 224-230
defining
programmatically, 226
XML resources, 226
loading, 227-228
simultaneously/sequentially, 226-227
transformations
alpha transparency, 228
defining, 224
interpolators, 230
moving, 229-230
rotating, 228-229
scaling, 229
TYPE_ACCELEROMETER sensor, 408
TYPE_GYROSCOPE sensor, 408
TYPE_LIGHT sensor, 409
TYPE_MAGNETIC_FIELD sensor, 409
TYPE_ORIENTATION sensor, 408
TYPE_PRESSURE sensor, 409
TYPE_PROXIMITY sensor, 409
TYPE_TEMPERATURE sensor, 409
unbindService() method, 446
underlining strings, 108
unexpected events, testing, 594
Uniform Resource Identifiers. See URIs
uninstalling applications, 651
update() method
   applications as content providers, 279-280
   SQLite databases, 242
updatering
   Android Market applications, 609
   App Widgets, 453, 454
      onUpdate() method, 458
      update service, creating, 458-459
   applications as content providers, 279-280
   best practices, 577-578
   content provider data, 268-269
   firmware upgrades, 569
   manifest files, 282
   notifications, 427-428
   preferences, 234
   SDK, 23, 31
   services, 442
   SQLite database records, 242-243
testing, 593
uploading applications to Android Market, 606-608
Uri class, 61
UriMatcher class, 277-278
URIs (Uniform Resource Identifiers), 25
   content, finding, 271
   defining, 276
   LiveFolders, defining, 283-284
   locations, mapping, 322
   pattern matching, 277-278
URL class, 288
URLs, querying, 289
URLUtil class, 307
usability testing, 592
USB drivers for Windows website, 67
use cases, developing, 555
UserDictionary content provider, 259, 267
user event handling. See event handling
user interfaces, 134
   adapters, 194
      arrays, 194-195
      binding data, 196
      cursor, 195-196
      event handling, 197
   buttons, 144
      basic, 144-146
      check boxes, 144, 146-147
      radio, 144, 148-149
      toggles, 144, 147
   clocks, 156-157
   compatibility
      Nine-Patch Stretchable images, 526-528
      screen support, 526
      working square principle, 528-531
   context menus, enabling, 159-161
database data, binding, 253-244
   adapter with ListView, 254-256
   adapters, 254
date input retrieval, 150-151
debugging, 180
designing, 572-573
dialogs, 165
   adding to activities, 166-167
   alert, 165
   character picker, 165
customizing, 168
date picker, 165
defining, 167
Dialog class, 165
dismissing, 167
initializing, 167
launching, 167
lifecycle, 166-167
progress, 165
removing, 167
time picker, 166
types, 165-166
documentation, 563
galleries, 194
grids, 194
layouts, creating, 134
    programmatically, 175-177
    XML resources, 173-175
lists, 194, 197-198
options menus, enabling, 157-159
progress indicators
    Chronometer class, 155-156
    ProgressBar class, 151-153
    RatingBar class, 154-155
    SeekBar class, 153-154
scrolling support, 201
sliding drawers, 202-203
styles. See styles
switchers, 202
tabs, 198
    creating from scratch, 200-201
    TabActivity class implementation, 198-200
text input retrieval
    EditText controls, 138-142
    Spinner controls, 143-144
    TextView, 134-138
    contextual links, creating, 136-138
    height, 136
text attribute, 135
width, 136
themes. See themes
time input retrieval, 151
view containers, 193
ViewGroups, 178
    child View objects, adding, 178
    layout classes, 178
    subclass categories, 178
    View container controls, 178
users
    applications as operating system users, 25
    billing, 611-612
        generation methods, 575
        testing, 594
    crash/bug reports, 569, 655-656
    demands, meeting, 572
    input retrieval
        EditText controls, 138-142
        Spinner controls, 143-144
<uses-configuration> tag
    input methods, 90
    trackballs, 519
<uses-feature> tag
    device features, configuring, 91
    GPS, 316
<uses-permission> tag, 94
<uses-sdk> tag, 88
utilities. See tools

V
versions
    applications, 86, 599
    compatibility, 546-548
        API levels, finding, 546-547
backward compatibility with Java Reflection, 547-548
  OpenGL ES, 368
  SDK, 87-90
    maximum, 90
    minimum, 89
    target, choosing, 89
  versioning systems, choosing, 564
vertical scrolling, 201
vertices (3D graphics)
  coloring, 377-378
  drawing, 376-377
vibration notifications, 429
video, 343
  formats, 351
  playing, 345-346
  recording, 343-345
  website, 351
Video.Media class, 260
VideoView widget, 345-346
View class, 133
  attributes, 127
  binding data. See adapters
  containers
    adapters, 194-197
    built-in, 193
    galleries, 194
    grids, 194
    lists, 194, 197-198
  scrolling support, 201
  sliding drawers, 202-203
  switchers, 202
  tabs. See tabs
drawing issues, debugging, 180
galleries, 178
Hierarchy Viewer, 179-180
drawing issues, debugging, 180
launching, 179
layout view, 180
pixel perfect view, 180-181
user interfaces, debugging, 180
list views, 178
OnFocusChangeListener interface, 164
OnLongClickListener interface, 164
parent-child relationships, 178
tab hosts, 178
themes, applying, 170
ViewWithRedDot subclass, 205-206
ViewGroups class, 178
  attributes, 182
  childView objects, adding, 178
  subclasses
    categories, 178
    layout classes, 178
    View container controls, 178
viewing
  application threads, 637-638
  live folder picker information, 484
  logs, 652
  network images, 295-297
  progress bars, 153
  retrieved images, 273-274
  styles, 128
  text, 134-135
  Web content, 302-304
ViewSwitcher class, 202
ViewTreeObserver class
  OnGlobalFocusChangeListener interface, 163
  OnGlobalLayoutListener, 163
  OnPreDrawListener interface, 163
  OnTouchModeChangeListener interface, 162
ViewWithRedDot class, 205-206
voice searches, 474-475
W

W3School's JavaScript Tutorial website, 314
WallpaperManager class, 342
wallpapers
  live, 461
    application support, 462
    creating, 462
    implementing services, 462
    installing, 465-466
    manifest file, configuring, 464-465
    service engine implementation, 463
    user events, handling, 463
    XML definition, 464
still images, 342-343
WallpaperService class, 462
<wallpaper> tag, 464
WAP (Wireless Application Protocol), 11-13
waterfall development, 552, 570
Web
  browsing, 301-302
    chrome, adding, 305-307
    event handling, 304-305
    Flash support, 311-313
    JavaScript, enabling, 304
    mouseovers, 304
    settings, configuring, 304
    zooming, 304
  content, loading, 302-304
  data, reading, 288-289
    errors, 289
    exception handling, 288
    permissions, 289
    URL class, 288
  layouts, designing, 302
  WebKit rendering engine, 301
    android.webkit package, 307
    classes, 307
    functionality, 308
    JavaScript interface application, 308-312
    Open Source Project website, 314
    support, 307
WebBackForwardList class, 307
WebChromeClient class, 305-307
WebHistoryItem class, 307
WebKit rendering engine, 301
  android.webkit package, 307
  classes, 307
  functionality, 308
  JavaScript interface application, 308-312
    Button control click handler, 311
    JavaScript control, 311
    JavaScript namespace, 309
    JavaScriptExtensions class, 309
    onCreate() method, 309
    sample.html file JavaScript functions, 310-311
    web page, defining, 310
    Open Source Project website, 314
    support, 307
WebSettings class, 304
websites
  3GPP Specifications, 365
  Activity class, 80
  ADB, 39
  Adobe AIR for Android
    beta program, 313
    Tool Suite, 314
  alternative resources, 549
  Android
    Dev Guide: “Developing on a Device” website, 67
    Development, 28, 398, 574
    sign-up, 604
Android Market, 612
  country requirements, 604
  Developer Distribution Agreement, 604
  help, 607
  licensing service, 604
Android.net package, 299
API levels, 96
ApiDemos application, 40
App Widgets, 454, 487
ash shell, 649
audio, 351
backup services, 497
backward compatibility without reflection, 548
best practices, 584
Bluetooth, 421
Borland SilkTest, 589
bug resolution process, 32
BusyBox, 660
C2DM (Cloud to Device Messaging), 438
chess font, 211
cloud computing Wikipedia, 497
compatibility, 549
ContactsContract content provider, 264
content providers, 285
Context class reference, 80
Cygwin, 398
DDMS, 38
Eclipse, 41
  download, 29
  IDE, 30
emulator, 38
Exerciser Monkey command-line tool, 596
extreme programming, 570
framework FAQ, 449
GNU
  Awk (Gawk) or Nawk, 398
  Make 3.81, 398
Google
  Android Developer's Guide, 41
  APIs Add-On, 35
  backup service, 492
  intents, 77
  Maps API key, 274, 333
IANA (Internet Assigned Numbers Authority), 467
input methods, 502
inputType attribute, 501
intent reference, 80
ISO 3166-1-alpha-2 Regions, 549
Issue Tracker, 32
Java
  JDK (Java Development Kit), 29
    JUnit, 582
    Platform, 41, 548
Java.net package, 299
Khronos OpenGL ES, 396
language support, 549
Licensing Agreement, 41
Linux Blog Man, 649
live folders, 487
live wallpapers, 487
locations and maps, 333
LunarLander application, 40
Manifest tag, 526
manifest files, 96
market filters, 599, 612
memory allocation, monitoring, 640
monkey tool, 659
multimedia formats, 351
multiple screen support, 96, 549
NDK, 405
Nexus One and Android Dev Phones, 570
NOAA: World Magnetic Model, 421
NotePad application, 40
NotificationManager class, 435
notifications, 435
Open Handset Alliance (OHA), 28
OpenGL ES, 396
OpenGL ES API documentation, 396
PhoneGap project, 311
processes and threads, 449
ProGuard, 611
Registry of Intents protocols, 77
resources, 132
runtime changes, 549
screen orientation changes, 522
SDK
documentation, 33-34
download, 29, 41
updates, 31
Searchable Configuration documentation, 475
searches, 487
security and permissions, 96
Sensor Simulator, 409
sensors, 421
services
lifecycle, 449
Service class, 449
Snake application, 40
SQLite, 258
SQLzoo.net, 258
Sync Adapter example application, 491, 497
system requirements, 29
testing references, 596
unit testing tutorial, 582
USB drivers for Windows, 67
video, 351
W3School’s JavaScript Tutorial, 314
WebKit
  Open Source Project, 314
  rendering engine, 301
Wikipedia
iterative development, 570
Rapid Application Development, 570
software process, 570
software testing, 596
waterfall development, 570
Write-Up, 365
Windows USB driver, 30
XML
  attributes for menus reference, 120
  Pull Parsing, 299
WebView class. See also WebKit rendering engine
benefits, 307
crime, adding, 305-307
content, loading, 302-304
event handling, 304-305
layouts, designing, 302
settings, configuring, 304
Web browsing, 301-302
WebViewClient class, 304-305
WHERE clauses (SQL queries), 247
white box testing, 591
widgets
MapView, 323-324
  Google Maps API Key, 325-326
  manifest file, 323
  MapController objects, 324
panning, 326-327  
permissions, 324  
points of interest, marking, 327-329  
MediaController, 345-346  
SurfaceView, 370-371  
VideoView, 345-346  

**WIP**  
access points, scanning, 412-413  
permissions, 412  
sensors, 412  
signal strength, 413  
testing, 414  

**WifiManager class, 412-413**  

**Wikipedia websites**  
iterative development, 570  
Rapid Application Development, 570  
software  
processes, 570  
testing, 596  
waterfall development, 570  
Write-Up, 365  

**windows (Eclipse IDE)**  
maximizing, 662  
minimizing, 662  
multiple file sections, viewing, 662  
open, limiting, 663  
side by side view, 662  

**Windows USB drivers, 30, 67**  

**Wireless Application Protocol (WAP), 11-13**  

**WML (Wireless Markup Language), 12**  

**working square principle, 528-531**  

**World Magnetic Model, 412**  

**WQVGA400 skin, 618**  

**WQVGA432 skin, 618**  

**WVGA800 skin, 618**  

**WVGA854 skin, 618**  

**X**  

**XML (Extensible Markup Language)**  
App Widget definitions, 453-454  
attributes, 120  
in-application search files, 471  
layouts  
accessing, 120-121, 126  
creating, 173-175  
live wallpaper definition, 464  
manifest files.  
See manifest files  
parsing, 290-291  
Pull Parsing website, 299  
services permissions file, 443  
shapes, defining, 214-215  
SMS permissions, 358  
tags.  
See tags  
telephony state information, 354  
tweened animations, defining, 226  
utility packages, 237  

**Z**  

**zooming**  
cameras, 341  
maps, 327  
Web browsing, 304