Version 6.9

Supports Through Android 6.0!

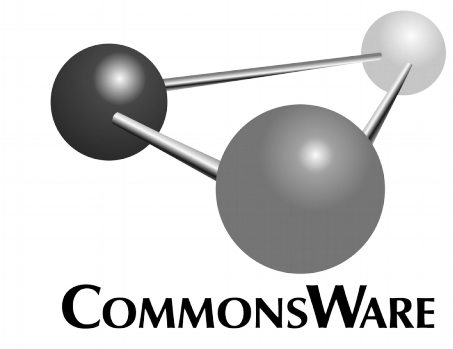
The Busy Coder's Guide to Android Development

Mark L. Murphy

CommonsWare

The Busy Coder's Guide to Android Development

by Mark L. Murphy



The Busy Coder's Guide to Android Development
by Mark L. Murphy

Printed in the United States of America.

Printing History:
October 2015: Version 6.9 ISBN: 978-0-9816780-0-9

Table of Contents

Headings formatted in ***bold-italic*** have changed since the last version.

- Preface
  - Welcome to the Book! i
  - ***The Book's Structure*** ..... ***i***
  - ***The Trails*** ..... ***ii***
  - ***About the Updates*** ..... ***vii***
  - ***Warescription*** ..... ***viii***
  - Book Bug Bounty ix
  - ***Source Code and Its License*** ..... ***ix***
  - ***Creative Commons and the Four-to-Free (42F) Guarantee*** ..... ***x***
  - Acknowledgments x
- Key Android Concepts
  - Android Applications 1
  - ***Android Devices*** ***7***
  - Don't Be Scared 10
- Choosing Your Development Toolchain
  - Android Studio 11
  - Eclipse 11
  - IntelliJ IDEA 12
  - ***Command-Line Builds via Gradle for Android*** ***12***
  - Yet Other Alternatives 12
  - IDEs... And This Book 13
  - What We Are Not Covering 13
- Tutorial #1 - Installing the Tools
  - Step #1 - Checking Your Hardware Requirements 15
  - Step #2 - Setting Up Java and 32-Bit Linux Support 16
  - ***Step #3 - Install the Developer Tools*** ***16***
  - ***Step #4 - Install the SDKs and Add-Ons*** ***17***
  - In Our Next Episode... 26
- Android and Projects
  - Common Concepts 27
  - ***Projects and Android Studio*** ***28***
  - Starter Project Generators 35
- Tutorial #2 - Creating a Stub Project
  - About Our Tutorial Project 37
  - About the Rest of the Tutorials 38

- About Our Tools 38
- ***Step #1: Creating the Project 38***
- ***Step #2 - Set Up the Emulator 40***
- Step #3 - Set Up the Device 46
- Step #4: Running the Project 49
- In Our Next Episode... 51

- Getting Around Android Studio
  - Navigating The Project Explorer 53
  - Running Projects 56
  - Viewing Output 57
  - Accessing Android Tools 58
  - Android Studio and Release Channels 62
  - Visit the Trails! 63
- Contents of Android Projects
  - ***What You Get, In General 65***
  - The Contents of an Android Studio Project 67
  - The Contents of an Eclipse Project 69
  - What You Get Out Of It 70
  - ***Projects and this Book 70***
- Introducing Gradle and the Manifest
  - ***Gradle: The Big Questions 71***
  - Obtaining Gradle 74
  - ***Versions of Gradle and Gradle for Android 76***
  - Gradle Environment Variables 77
  - ***Examining the Gradle Files 77***
  - Introducing the Manifest 79
  - Things In Common Between the Manifest and Gradle 80
  - Other Gradle Items of Note 83
  - Where's the GUI? 85
  - The Rest of the Manifest 85
  - Learning More About Gradle 87
  - Visit the Trails! 88
- Tutorial #3 - Changing Our Manifest (and Gradle File)
  - Some Notes About Relative Paths 89
  - Step #1: Supporting Screens 90
  - Step #2: Adding our Minimum and Target SDK Versions 90
  - In Our Next Episode... 91
- Some Words About Resources
  - String Theory 94
  - Got the Picture? 98
  - Dimensions 103

- The Resource That Shall Not Be Named... Yet 105
- Tutorial #4 - Adjusting Our Resources
  - Step #1: Changing the Name 107
  - Step #2: Changing the Icon 108
  - Step #3: Running the Result 111
  - In Our Next Episode... 113
- The Theory of Widgets
  - What Are Widgets? 115
  - Size, Margins, and Padding 117
  - What Are Containers? 117
  - The Absolute Positioning Anti-Pattern 118
  - The Theme of This Section: Themes 119
- The Android User Interface
  - The Activity 123
  - Dissecting the Activity 124
  - Using XML-Based Layouts 125
- Basic Widgets
  - Common Concepts 131
  - Assigning Labels 134
  - A Commanding Button 144
  - Fleeting Images 149
  - Fields of Green. Or Other Colors. 156
  - More Common Concepts 162
  - Visit the Trails! 164
- Debugging Your App
  - Get Thee To a Stack Trace 166
  - The Case of the Confounding Class Cast 170
  - Point Break 171
- LinearLayout and the Box Model
  - Concepts and Properties 173
  - Android Studio Graphical Layout Editor 177
  - Eclipse Graphical Layout Editor 178
- Other Common Widgets and Containers
  - Just a Box to Check 181
  - Don't Like Checkboxes? How About Toggles or Switches? 186
  - Turn the Radio Up 191
  - All Things Are Relative 195
  - Tabula Rasa 202
  - Scrollwork 209
  - Making Progress with ProgressBars 212
  - Visit the Trails! 213

- Tutorial #5 - Making Progress
  - Step #1: Removing The "Hello, World" 215
  - Step #2: Adding a ProgressBar 216
  - Step #3: Seeing the Results 217
  - In Our Next Episode... 218
- GUI Building, Continued
  - Making Your Selection 219
  - Including Includes 220
  - Wrap It Up (In a Container) 222
  - Morphing Widgets 223
  - Preview of Coming Attractions 224
- AdapterViews and Adapters
  - Adapting to the Circumstances 225
  - Lists of Naughty and Nice 227
  - Clicks versus Selections 229
  - Spin Control 233
  - Grid Your Lions (Or Something Like That...) 236
  - Fields: Now With 35% Less Typing! 239
  - Customizing the Adapter 243
  - Visit the Trails! 251
- The WebView Widget
  - Role of WebView 253
  - WebView and WebKit 254
  - The Android System WebView 254
  - Adding the Widget 255
  - Loading Content Via a URL 255
  - Links and Redirects 257
  - Supporting JavaScript 257
  - Alternatives for Loading Content 258
  - Listening for Events 259
  - Addressing the Link/Redirect Behavior 262
  - Visit the Trails! 263
- Defining and Using Styles
  - Styles: DIY DRY 265
  - Elements of Style 267
  - Themes: Would a Style By Any Other Name... 270
  - What Happens If You Have No Theme 270
- JARs and Library Projects
  - The Dalvik VM, and a Bit of ART 274
  - Getting the Library 274
  - The Outer Limits 276

- JAR Dependency Management 277
- OK, So What is a Library Project? 277
- Using a Library Project 278
- Library Projects: What You Get 279
- ***The Android Support Package 280***

- Tutorial #6 - Adding a Library
  - Step #1: Attaching the Android Support Package 285
  - ***Step #2: Attaching the Third-Party Dependencies 286***
  - In Our Next Episode... 287
- The Action Bar
  - Bar Hopping 289
  - Yet Another History Lesson 295
  - Your Action Bar Options 296
  - Setting the Target 298
  - Defining the Resource 299
  - Applying the Resource 301
  - Responding to Events 302
  - The Rest of the Sample Activity 303
  - MENU Key, We Hardly Knew Ye 309
  - Action Bars, Live in Living Color! 310
  - Visit the Trails! 319
- Tutorial #7 - Setting Up the Action Bar
  - Step #1: Acquiring Some Icons 321
  - Step #2: Defining Some Options 323
  - Step #3: Loading and Responding to Our Options 324
  - Step #4: Trying It Out 325
  - In Our Next Episode... 328
- Android's Process Model
  - When Processes Are Created 329
  - BACK, HOME, and Your Process 330
  - Termination 331
  - Foreground Means "I Love You" 332
  - You and Your Heap 332
- Activities and Their Lifecycles
  - Creating Your Second (and Third and...) Activity 334
  - Warning! Contains Explicit Intents! 340
  - Using Implicit Intents 342
  - Extra! Extra! 347
  - Pondering Parcelable 349
  - Asynchronicity and Results 350
  - Schroedinger's Activity 350

- Life, Death, and Your Activity 351
- When Activities Die 353
- Walking Through the Lifecycle 354
- Recycling Activities 357
- Application: Transcending the Activity 358

- Tutorial #8 - Setting Up An Activity
  - ***Step #1: Creating the Stub Activity Class and Manifest Entry 361***
  - Step #2: Launching Our Activity 363
  - In Our Next Episode... 363

- The Tactics of Fragments
  - The Six Questions 365
  - Where You Get Your Fragments From 368
  - Your First Fragment 368
  - The Fragment Lifecycle Methods 371
  - Your First Dynamic Fragment 372
  - Fragments and the Action Bar 375
  - Fragments Within Fragments: Just Say "Maybe" 376
  - Fragments and Multiple Activities 377

- Tutorial #9 - Starting Our Fragments
  - Step #1: Create a SimpleContentFragment 379
  - Step #2: Examining SimpleContentFragment 381
  - In Our Next Episode... 382

- Swiping with ViewPager
  - Swiping Design Patterns 383
  - Pieces of a Pager 384
  - Paging Fragments 384
  - Paging Other Stuff 389
  - Indicators 390

- Tutorial #10 - Rigging Up a ViewPager
  - Step #1: Add a ViewPager to the Layout 395
  - Step #2: Obtaining Our ViewPager 396
  - Step #3: Creating a ContentsAdapter 397
  - ***Step #4: Setting Up the ViewPager 397***
  - In Our Next Episode... 400

- Resource Sets and Configurations
  - What's a Configuration? And How Do They Change? 401
  - Configurations and Resource Sets 402
  - Screen Size and Orientation 403
  - Coping with Complexity 406
  - Choosing The Right Resource 407
  - API-Versioned Resources 411

- Default Change Behavior 413
- State Saving Scenarios 414
- Your Options for Configuration Changes 415
- Blocking Rotations 427
- And Now, a Word From the Android Project View 427

- Material Design Basics
  - Your App, in Technicolor! 431
- Dealing with Threads
  - The Main Application Thread 439
  - Getting to the Background 441
  - Asyncing Feeling 441
  - Alternatives to AsyncTask 450
  - And Now, The Caveats 451
  - Event Buses 452
  - Visit the Trails! 459
- Requesting Permissions
  - ***Frequently-Asked Questions About Permissions 462***
  - ***Characteristics of Permissions 467***
  - ***New Permissions in Old Applications 470***
  - ***Android 6.0+ Runtime Permission System 471***
- Tutorial: Runtime Permission Support
  - Step #0: Install the Android 6.0 SDK 484
  - ***Step #1: Import and Review the Starter Project 485***
  - Step #2: Update Gradle for Android 6.0 492
  - Step #3: Review the Planned UX 494
  - ***Step #4: Detect the First Run 495***
  - ***Step #5: On First Run, Ask For Permissions 496***
  - ***Step #6: Check for Permissions Before Taking a Picture 500***
  - ***Step #7: Detect If We Should Show Some Rationale 501***
  - ***Step #8: Add a Rationale UI and Re-Request Permissions 503***
  - Step #9: Check for Permissions Before Recording a Video 508
  - ***Step #10: Detect If We Should Show Some Rationale (Again) 509***
  - ***Step #11: Support Configuration Changes 511***
- Assets, Files, and Data Parsing
  - Packaging Files with Your App 521
  - Files and Android 523
  - Working with Internal Storage 524
  - ***Working with External Storage 532***
  - Multiple User Accounts 537
  - Linux Filesystems: You Sync, You Win 538
  - ***StrictMode: Avoiding Janky Code 540***

- XML Parsing Options 542
- JSON Parsing Options 543
- Visit the Trails! 543

- Tutorial #11 - Adding Simple Content
  - Step #1: Adding Some Content 545
  - Step #2: Using SimpleContentFragment 546
  - Step #3: Launching Our Activities, For Real This Time 547
  - Step #4: Getting a Bit More Material 548
  - Step #5: Seeing the Results 549
  - In Our Next Episode... 552
- Tutorial #12 - Displaying the Book
  - Step #1: Adding a Book 553
  - Step #2: Creating a ModelFragment 554
  - Step #3: Defining Our Model 555
  - Step #4: Examining Our Model 557
  - Step #5: Defining Our Event 557
  - Step #6: Loading Our Model 558
  - Step #7: Registering for Events 560
  - Step #8: Adapting the Content 561
  - Step #9: Showing the Content When Loaded 562
  - Step #10: Attaching our ModelFragment 562
  - Step #11: Showing the Content After a Configuration Change 564
  - Step #12: Setting Up StrictMode 564
  - In Our Next Episode... 565
- Using Preferences
  - Getting What You Want 567
  - Stating Your Preference 568
  - Collecting Preferences with PreferenceFragment 569
  - Types of Preferences 576
- Tutorial #13 - Using Some Preferences
  - Step #1: Defining the Preference XML Files 583
  - Step #2: Creating Our Preference Activity 584
  - Step #3: Adding To Our Action Bar 585
  - Step #4: Launching the Preference Activity 586
  - Step #5: Loading the Preferences 588
  - Step #6: Saving the Last-Read Position 591
  - Step #7: Restoring the Last-Read Position 592
  - Step #8: Keeping the Screen On 593
  - In Our Next Episode... 595
- SQLite Databases
  - Introducing SQLite 597

- Thinking About Schemas 598
- Start with a Helper 598
- Getting Data Out 603
- The Rest of the CRUD 609
- Hey, What About Hibernate? 615
- Visit the Trails! 615

- Tutorial #14 - Saving Notes
  - Step #1: Adding a DatabaseHelper 617
  - Step #2: Examining DatabaseHelper 618
  - Step #3: Creating a NoteFragment 619
  - Step #4: Examining NoteFragment 622
  - Step #5: Creating the NoteActivity 623
  - Step #6: Examining NoteActivity 624
  - Step #7: Add Notes to the Action Bar 624
  - Step #8: Defining a NoteLoadedEvent 625
  - Step #9: Loading a Note from the Database 626
  - Step #10: Loading the Note Into the Fragment 627
  - Step #11: Updating the Database 628
  - Step #12: Saving the Note 629
  - Step #13: Adding a Delete Action Bar Item 631
  - Step #14: Closing the NoteFragment When Deleted 632
  - In Our Next Episode... 637
- Internet Access
  - DIY HTTP 639
  - HTTP via DownloadManager 649
  - Using Third-Party JARs 649
  - SSL 650
  - Using HTTP Client Libraries 650
  - ***Visit the Trails 663***
- Intents, Intent Filters
  - What's Your Intent? 665
  - Stating Your Intent(ions) 667
  - Responding to Implicit Intents 667
  - Requesting Implicit Intents 670
  - ShareActionProvider 674
- Broadcasts and Broadcast Receivers
  - The Stopped State 679
  - Example System Broadcasts 680
  - The Order of Things 688
  - Keeping It Local 689
  - Visit the Trails! 690

- Tutorial #15 - Sharing Your Notes
  - Step #1: Adding a ShareActionProvider 691
  - Step #2: Sharing the Note 692
  - Step #3: Testing the Result 694
  - In Our Next Episode... 696
- Services and the Command Pattern
  - Why Services? 697
  - Setting Up a Service 698
  - Communicating To Services 700
  - Scenario: The Music Player 702
  - Communicating From Services 705
  - Scenario: The Downloader 707
  - Services and Configuration Changes 712
- Tutorial #16 - Updating the Book
  - Step #1: Adding a Stub DownloadCheckService 713
  - Step #2: Tying the Service Into the Action Bar 714
  - Step #3: Defining Our Event 715
  - Step #4: Defining Our JSON 715
  - Step #5: Defining Our Retrofit Interface 716
  - Step #6: Retrieving Our JSON Via Retrofit 717
  - Step #7: Downloading the Update 718
  - Step #8: Unpacking the Update 720
  - Step #9: Using the Update 722
  - In Our Next Episode... 727
- Large-Screen Strategies and Tactics
  - Objective: Maximum Gain, Minimum Pain 729
  - The Fragment Strategy 729
  - Fragment Example: The List-and-Detail Pattern 738
  - Other Master-Detail Strategies 750
  - Showing More Pages 762
  - Fragment FAQs 767
  - Screen Size and Density Tactics 768
  - Other Considerations 772
- Tutorial #17 - Supporting Large Screens
  - Step #1: Creating Our Layouts 775
  - Step #2: Loading Our Sidebar Widgets 779
  - Step #3: Opening the Sidebar 779
  - Step #4: Loading Content Into the Sidebar 780
  - Step #5: Removing Content From the Sidebar 782
- Backwards Compatibility Strategies and Tactics
  - Think Forwards, Not Backwards 789

- Aim Where You Are Going 791
- A Target-Rich Environment 791
- Lint: It's Not Just For Belly Buttons 792
- A Little Help From Your Friends 793
- Avoid the New on the Old 793
- Testing 797
- Keeping Track of Changes 797

- Google Play Services
  - ***What Is Google Play Services?* *799***
  - ***What Is In the Play Services SDK?* *800***
  - ***Adding Play Services to Your Project* *809***
- Getting Help
  - Questions. Sometimes, With Answers. 819
  - Heading to the Source 820
  - Getting Your News Fix 821
- Working with Library Projects
  - Prerequisites 823
  - ***Creating a Library Project* *823***
  - Using a Library Project, Part II 827
  - Library Projects and the Manifest 827
  - Limitations of Library Projects 828
- Gradle and Eclipse Projects
  - ***Prerequisites and Warnings* *829***
  - "Legacy"? 829
  - ***Creating Your Gradle Build File* *830***
  - Examining the Gradle File 835
- Gradle and Tasks
  - Key Build-Related Tasks 837
  - Results 839
- Gradle and the New Project Structure
  - ***Prerequisites and Warnings* *841***
  - ***Objectives of the New Project Structure* *842***
  - ***Terminology* *842***
  - ***Creating a Project in the New Structure* *846***
  - ***What the New Project Structure Looks Like* *846***
  - Configuring the Stock Build Types 849
  - Adding Build Types 854
  - Adding Product Flavors and Getting Build Variants 855
  - ***Doing the Splits* *860***
  - Revisiting the Legacy Gradle File 862
  - Working with the New Project Structure in Android Studio 863

- Flavors, Build Types, and the Project Structure Dialog 866
- Gradle and Dependencies
  - ***Prerequisites and Warnings 867***
  - "Dependencies"? 868
  - ***A Tale of Two Dependencies Closures 868***
  - Depending Upon a JAR 868
  - Depending Upon NDK Binaries 870
  - Depending Upon an Android Library Project 870
  - Depending Upon Sub-Projects 872
  - Depending Upon Artifacts 873
  - Creating Android JARs from Gradle 880
  - A Property of Transitive (Dependencies) 881
  - Dependencies By Build Type 882
  - Dependencies By Flavor 882
  - ***Examining Some CWAC Builds 883***
  - Dependencies and the Project Structure Dialog 889
- Manifest Merger Rules
  - ***Prerequisites 891***
  - Manifest Scenarios 892
  - Pieces of Manifest Generation 893
  - Examining the Merger Results 895
  - ***Merging Elements and Attributes 896***
  - ***Employing Placeholders 902***
- Signing Your App
  - Prerequisites 907
  - Role of Code Signing 907
  - What Happens In Debug Mode 908
  - Production Signing Keys 909
- Distribution
  - Prerequisites 925
  - Get Ready To Go To Market 925
- Advanced Gradle for Android Tips
  - ***Prerequisites 931***
  - Gradle, DRY 931
  - Automating APK Version Information 937
  - ***Adding to BuildConfig 939***
  - ***Down and Dirty with the DSL 941***
- JUnit and Android
  - Prerequisites 943
  - JUnit Basics 943
  - Pondering Gradle for Android 944

- Where Your Test Code Lives 945
- Where Your Test Code Runs 949
- Writing Your Test Cases 950
- Your Test Suite 955
- Running Your Tests 955
- Testing Android Library Projects 962
- Test Dependencies 963
- Testing Legacy Project Structures with Gradle for Android 964

- Testing with JUnit4
  - Prerequisites 967
  - ***The AndroidJUnitRunner 967***
  - JUnit4 970
- MonkeyRunner and the Test Monkey
  - Prerequisites 979
  - MonkeyRunner 979
  - Monkeying Around 981
- Testing with UI Automator
  - Prerequisites 985
  - What Is UI Automator? 985
  - Why Choose UI Automator Over Alternatives? 986
  - Gradle and Android Studio Settings 986
  - Creating a Test Case 987
  - Running Your Tests 994
  - Finding Your Widgets 994
  - Using the UI Automator Viewer 995
- Notifications
  - What's a Notification? 999
  - Showing a Simple Notification 1002
  - The Activity-Or-Notification Scenario 1006
  - Big (and Rich) Notifications 1007
  - Foreground Services 1014
  - Disabled Notifications 1017
- Advanced Notifications
  - Prerequisites 1021
  - Being a Good Citizen 1021
  - Wear? There! 1022
  - Stacking Notifications 1027
  - Avoiding Wear 1033
  - Other Wear-Specific Notification Options 1034
  - Lockscreen Notifications 1047
  - Priority, and Heads-Up Notifications 1056

- Full-Screen Notifications 1058
- Custom Views 1061
- Seeing It In Action 1062
- Progress Notifications 1068
- Life After Delete 1071
- The Mysterious Case of the Missing Number 1072

- Introducing GridLayout
  - Prerequisites 1073
  - Issues with the Classic Containers 1073
  - The New Contender: GridLayout 1075
  - GridLayout and the Android Support Package 1075
  - Eclipse and GridLayout 1077
  - Trying to Have Some Rhythm 1077
  - Our Test App 1078
  - Replacing the Classics 1080
  - Implicit Rows and Columns 1086
  - Row and Column Spans 1088
- Dialogs and DialogFragments
  - Prerequisites 1093
  - DatePickerDialog and TimePickerDialog 1093
  - AlertDialog 1099
  - DialogFragments 1100
  - DialogFragment: The Other Flavor 1104
  - Dialogs: Modal, Not Blocking 1105
- Advanced ListViews
  - Prerequisites 1107
  - Multiple Row Types, and Self Inflation 1107
  - Choice Modes and the Activated Style 1113
  - Custom Mutable Row Contents 1114
  - From Head To Toe 1119
  - Enter RecyclerView 1123
- Action Bar Navigation
  - Prerequisites 1125
  - List Navigation 1125
  - Tabs (And Sometimes List) Navigation 1130
  - Custom Navigation 1136
- Action Modes
  - Prerequisites 1138
  - A Matter of Context 1138
  - Manual Action Modes 1139
  - Multiple-Choice-Modal Action Modes 1144

- Long-Click To Initiate an Action Mode 1148
- Other Advanced Action Bar Techniques
  - Prerequisites 1155
  - Action Layouts 1155
  - Action Views and Action Providers 1156
  - Searching with SearchView 1157
  - Floating Action Bars 1162
- Toolbar
  - Prerequisites 1167
  - Basic Toolbar Mechanics 1167
  - Use Case #1: Split Action Bar 1168
  - Use Case #2: Contextual Actions 1173
  - Use Case #3: Replacement Action Bar 1182
- AppCompat: The Official Action Bar Backport
  - Prerequisites 1185
  - Ummmm... Why? 1185
  - The Basics of Using AppCompat 1188
  - Other AppCompat Effects 1194
  - ***Toolbar and AppCompat 1199***
  - To Material, or Not to Material 1202
- ActionBarSherlock
  - Prerequisites 1205
  - Installation 1206
  - Code Changes 1206
- RecyclerView
  - Prerequisites 1210
  - AdapterView and its Discontents 1210
  - Enter RecyclerView 1210
  - A Trivial List 1211
  - Divider Options 1219
  - Handling Click Events 1225
  - What About Cursors? 1231
  - Grids 1235
  - Varying the Items 1238
  - Mutable Row Contents 1250
  - Changing the Contents 1270
  - The Order of Things 1275
  - Other Bits of Goodness 1283
  - The March of the Libraries 1284
- Implementing a Navigation Drawer
  - Prerequisites 1287

- What is a Navigation Drawer? 1287
- A Simple Navigation Drawer 1289
- Alternative Row Layouts 1295
- Additional Considerations 1297
- What Should Not Be in the Drawer 1306
- Independent Implementations 1306

- The Android Design Support Library
  - Prerequisites 1308
  - GUIs and the Support Package 1308
  - Adding the Library... and What Comes With It 1309
  - Some Notes About Icons 1309
  - ***Snackbars: Sweeter than Toasts 1309***
  - ***Absolutely FABulous 1315***
  - Material Tabs with TabLayout 1331
  - Floating Labels 1341
- Advanced Uses of WebView
  - Prerequisites 1351
  - Friends with Benefits 1351
  - Turnabout is Fair Play 1357
  - Navigating the Waters 1361
  - Settings, Preferences, and Options (Oh, My!) 1361
  - Security and Your WebView 1362
- The Input Method Framework
  - Prerequisites 1367
  - Keyboards, Hard and Soft 1367
  - Tailored To Your Needs 1368
  - Tell Android Where It Can Go 1373
  - Fitting In 1375
  - Jane, Stop This Crazy Thing! 1377
- Fonts
  - ***Prerequisites 1379***
  - Love The One You're With 1379
  - Yeah, But Do We Really Have To Do This in Java? 1383
  - Here a Glyph, There a Glyph 1384
- Rich Text
  - Prerequisites 1387
  - The Span Concept 1387
  - Loading Rich Text 1389
  - Editing Rich Text 1392
  - Saving Rich Text 1396
  - Manipulating Rich Text 1397

- Animators
  - Prerequisites 1399
  - ***ViewPropertyAnimator* 1399**
  - The Foundation: Value and Object Animators 1404
  - Animating Custom Types 1406
  - Hardware Acceleration 1408
  - The Three-Fragment Problem 1409
- Legacy Animations
  - Prerequisites 1421
  - It's Not Just For Toons Anymore 1421
  - A Quirky Translation 1422
  - Fading To Black. Or Some Other Color. 1426
  - When It's All Said And Done 1428
  - Loose Fill 1429
  - Hit The Accelerator 1429
  - Animate. Set. Match. 1430
  - Active Animations 1431
- Custom Drawables
  - Prerequisites 1433
  - Where Do These Things Go? 1434
  - ColorDrawable 1434
  - AnimationDrawable 1434
  - StateListDrawable 1438
  - ColorStateList 1439
  - LayerDrawable 1441
  - TransitionDrawable 1442
  - LevelListDrawable 1443
  - ScaleDrawable and ClipDrawable 1444
  - InsetDrawable 1453
  - ShapeDrawable 1454
  - VectorDrawable 1464
  - BitmapDrawable 1464
  - Composite Drawables 1471
  - A Stitch In Time Saves Nine 1474
- Mapping with Maps V2
  - Prerequisites 1485
  - A Brief History of Mapping on Android 1486
  - Where You Can Use Maps V2 1486
  - Licensing Terms for Maps V2 1487
  - ***What You Need to Start* 1487**
  - The Book Samples... And You! 1491

- ***Setting Up a Basic Map 1492***
- Playing with the Map 1497
- Map Tiles 1500
- Placing Simple Markers 1500
- Seeing All the Markers 1503
- Flattening and Rotating Markers 1505
- Sprucing Up Your "Info Windows" 1509
- Images and Your Info Window 1513
- Setting the Marker Icon 1519
- Responding to Taps 1521
- Dragging Markers 1522
- The "Final" Limitations 1524
- A Bit More About IPC 1527
- ***Finding the User 1528***
- ***Dealing with Runtime Permissions 1533***
- Drawing Lines and Areas 1535
- Gestures and Controls 1538
- Tracking Camera Changes 1539
- Maps in Fragments and Pagers 1541
- Animating Marker Movement 1545
- Maps, of the Indoor Variety 1555
- Taking a Snapshot of a Map 1555
- MapFragment vs. MapView 1556
- ***About That AbstractMapActivity Class... ..... 1557***
- Helper Libraries for Maps V2 1561
- Problems with Maps V2 at Runtime 1565
- Problems with Maps V2 Deployment 1565
- What Non-Compliant Devices Show 1565
- Mapping Alternatives 1566
- News and Getting Help 1566

- Crafting Your Own Views
  - Prerequisites 1569
  - Pick Your Poison 1569
  - Colors, Mixed How You Like Them 1570
  - ReverseChronometer: Simply a Custom Subclass 1580
  - AspectLockedFrameLayout: A Custom Container 1586
  - Mirror and MirroringFrameLayout: Draw It Yourself 1589
- Advanced Preferences
  - Prerequisites 1601
  - Introducing PreferenceActivity 1601
  - Intents for Headers or Preferences 1605

- Conditional Headers 1606
- Dependent Preferences 1611
- Nested Screens 1613
- ***Listening to Preference Changes 1616***
- ***Defaults, and Defaults 1619***
- Listening to Preference Value Changes 1620

- Custom Dialogs and Preferences
  - Prerequisites 1621
  - Your Dialog, Chocolate-Covered 1621
  - Preferring Your Own Preferences, Preferably 1625
- Progress Indicators
  - Prerequisites 1631
  - Progress Bars 1631
  - ProgressBar and Threads 1635
  - ***Tailoring Progress Bars 1637***
  - Progress Dialogs 1647
  - ***Title Bar and Action Bar Progress Indicators 1649***
  - Direct Progress Indication 1651
- More Fun with Pagers
  - Prerequisites 1653
  - Hosting ViewPager in a Fragment 1653
  - Pages and the Action Bar 1655
  - ViewPagers and Scrollable Contents 1657
  - Using ViewPagerIndicator 1658
  - Columns for Large, Pages for Small 1662
  - Introducing ArrayPagerAdapter 1668
  - Columns for Large Landscape, Pages for the Rest 1671
  - Adding, Removing, and Moving Pages 1676
  - Inside ArrayPagerAdapter 1680
- Focus Management and Accessibility
  - Prerequisites 1693
  - Prepping for Testing 1694
  - Controlling the Focus 1694
  - Accessibility and Focus 1703
  - Accessibility Beyond Focus 1703
  - Accessibility Beyond Impairment 1713
- Miscellaneous UI Tricks
  - Prerequisites 1715
  - Full-Screen and Lights-Out Modes 1715
  - Offering a Delayed Timeout 1726
- Event Bus Alternatives

- Prerequisites 1731
- A Brief Note About the Sample Apps 1731
- Standard Intents as Event Bus 1731
- LocalBroadcastManager as Event Bus 1732
- Square's Otto 1742
- Revisiting greenrobot's EventBus 1748

- Tasks
  - Prerequisites 1753
  - First, Some Terminology 1753
  - And Now, a Bit About Task Killers 1759
  - A Canary for the Task's Coal Mine 1766
  - The Default User Experience 1768
  - Explaining the Default Behavior 1772
  - Basic Scenarios for Changing the Behavior 1776
  - Dealing with the Persistent Tasks 1789
  - Documents As Tasks 1791
  - Other Task-Related Activity Properties 1795
  - Other Task-Related Activity Methods 1803
- The Assist API ("Now On Tap")
  - Prerequisites 1806
  - ***What Data Gets Disclosed 1806***
  - ***Adding to the Data 1808***
  - ***Removing from the Data 1810***
  - ***Blocking Assist as a User 1813***
  - ***Implementing Your Own Assistant 1816***
- Home Screen App Widgets
  - Prerequisites 1827
  - App Widgets and Security 1827
  - The Big Picture for a Small App Widget 1828
  - Crafting App Widgets 1829
  - Another and Another 1836
  - App Widgets: Their Life and Times 1836
  - Controlling Your (App Widget's) Destiny 1837
  - One Size May Not Fit All 1837
  - Lockscreen Widgets 1844
  - ***Preview Images 1850***
  - Being a Good Host 1852
- Adapter-Based App Widgets
  - Prerequisites 1853
  - ***AdapterViews for App Widgets 1853***
  - ***Building Adapter-Based App Widgets 1854***

- Content Provider Theory
  - Prerequisites 1869
  - Using a Content Provider 1869
  - Building Content Providers 1877
  - Issues with Content Providers 1885
- Content Provider Implementation Patterns
  - Prerequisites 1887
  - The Single-Table Database-Backed Content Provider 1887
  - The Local-File Content Provider 1894
  - The Protected Provider 1902
  - The Stream Provider 1904
  - FileProvider 1907
  - StreamProvider 1912
- The Loader Framework
  - Prerequisites 1915
  - Cursors: Issues with Management 1916
  - Introducing the Loader Framework 1916
  - Honeycomb... Or Not 1918
  - Using CursorLoader 1918
  - What Else Is Missing? 1920
  - Issues, Issues, Issues 1921
  - Loaders Beyond Cursors 1921
  - What Happens When...? 1921
- The ContactsContract and CallLog Providers
  - Prerequisites 1925
  - Introducing You to Your Contacts 1926
  - Pick a Peck of Pickled People 1927
  - Spin Through Your Contacts 1929
  - Makin' Contacts 1934
  - Looking at the CallLog 1938
- The CalendarContract Provider
  - Prerequisites 1944
  - You Can't Be a Faker 1944
  - Do You Have Room on Your Calendar? 1944
  - Penciling In an Event 1949
- The MediaStore Provider
  - ***Prerequisites 1951***
  - What Is the MediaStore? 1952
  - MediaStore and "Other" External Storage 1953
  - How Does My Content Get Indexed? 1954
  - How Do I Retrieve Video from the MediaStore? 1954

- Consuming Documents
  - Prerequisites 1963
  - The Storage Access... What? 1963
  - The Storage Access Framework Participants 1965
  - Picking How to Pick (a Peck of Pickled Pepper Photos) 1965
  - Opening a Document 1966
  - The Rest of the CRUD 1969
  - Pondering Persistent Permissions 1970
- Providing Documents
  - Prerequisites 1973
  - Have Your Content, and Provide it Too 1973
  - Key Provider Concepts 1975
  - Pieces of a Provider 1976
  - Optional Provider Capabilities 1990
- Encrypted Storage
  - Prerequisites 1998
  - Scenarios for Encryption 1998
  - ***Obtaining SQLCipher 1998***
  - Using SQLCipher 1999
  - SQLCipher Limitations 2000
  - Passwords and Sessions 2001
  - About Those Passphrases... 2002
  - Encrypted Preferences 2009
  - IOCipher 2011
- Packaging and Distributing Data
  - Prerequisites 2013
  - Packing a Database To Go 2013
- Advanced Database Techniques
  - Prerequisites 2017
  - Full-Text Indexing 2017
- Embedding a Web Server
  - ***Prerequisites 2035***
  - ***Why a Web Server? 2035***
  - ***Introducing AsyncHttpServer 2037***
  - ***Embedding a Simple Server 2038***
  - ***Template-Driven Responses, with Handlebars 2053***
  - ***Supporting WebSockets 2058***
  - ***Securing the Web Server 2063***
  - ***Towards a Reusable Web Server Service 2070***
  - Prerequisites 2080
  - ***Downloading Files 2080***

- Audio Playback
  - Prerequisites 2095
  - Get Your Media On 2095
  - MediaPlayer for Audio 2096
  - Other Ways to Make Noise 2101
- Audio Recording
  - Prerequisites 2105
  - Recording by Intent 2105
  - Recording to Files 2108
  - Recording to Streams 2111
  - Raw Audio Input 2114
  - Requesting the Microphone 2114
- Video Playback
  - Prerequisites 2115
  - Moving Pictures 2115
- Using the Camera via 3rd-Party Apps
  - Prerequisites 2121
  - Being Specific About Features 2121
  - ***Still Photos: Letting the Camera App Do It 2122***
  - Scanning with ZXing 2125
  - Videos: Letting the Camera App Do It 2126
  - CWAC-Cam2: A CameraActivity Of Your Own 2128
  - Directly Working with the Camera 2135
- Working Directly with the Camera
  - Prerequisites 2137
  - Notes About the Code Snippets 2137
  - A Tale of Two APIs 2138
  - Performing Basic Camera Operations 2139
  - And Now, The Problems 2161
- Media Routes
  - Prerequisites 2167
  - Terminology 2167
  - ***A Tale of Two MediaRouters 2168***
  - Attaching to MediaRouter 2169
  - ***User Route Selection with MediaRouteActionProvider 2171***
  - Using Live Video Routes 2186
  - Using Remote Playback Routes 2186
- Supporting External Displays
  - Prerequisites 2205
  - A History of External Displays 2205
  - What is a Presentation? 2206

- Playing with External Displays 2207
- Detecting Displays 2213
- A Simple Presentation 2214
- ***A Simpler Presentation 2220***
- Presentations and Configuration Changes 2225
- Presentations as Fragments 2226
- Another Sample Project: Slides 2236
- Device Support for Presentation 2243
- Presentations from a Service 2243
- Hey, What About Chromecast? 2246

- Google Cast and Chromecast
  - Prerequisites 2249
  - Here a Cast, There a Cast 2249
  - Common Chromecast Development Notes 2251
  - Your API Choices 2251
  - Senders and Receivers 2252
  - Supported Media Types 2253
  - ***Cast SDK Dependencies 2254***
  - Developing Google Cast Apps 2256
- The "Ten-Foot UI"
  - Prerequisites 2257
  - What is the "Ten-Foot UI"? 2258
  - Overscan 2258
  - Navigation 2259
  - Stylistic Considerations 2260
  - The Leanback UI 2262
  - Testing Your Theories 2276
- Putting the TVs All Together: Decktastic
  - Prerequisites 2280
  - Introducing Decktastic 2280
  - Trying Decktastic Yourself 2284
  - ***Implementing Decktastic 2284***
- Creating a MediaRouteProvider
  - Prerequisites 2315
  - Terminology 2315
  - DIY Chromecast 2316
  - ***Creating the MediaRouteProvider 2318***
  - Consuming the MediaRouteProvider 2328
  - ***Implementing This "For Realz" 2331***
- Screenshots and Screen Recordings
  - ***Prerequisites 2335***

- ***Requesting Screenshots*** ***2335***
- ***Recording the Screen*** ***2349***

- SSL
  - Prerequisites 2351
  - Basic SSL Operation 2351
  - Common SSL Problems 2352
  - TrustManagers 2353
  - TrustManagerBuilder 2355
  - About That Man in the Middle 2359
  - Self-Signed Certificates, Revisited 2361
  - Certificate Memorizing 2361
  - Pinning 2366
  - NetCipher 2366
- Advanced Permissions
  - Prerequisites 2369
  - Securing Yourself 2369
  - Signature Permissions 2372
  - The Custom Permission Vulnerability 2373
- Restricted Profiles and UserManager
  - ***Prerequisites*** ***2385***
  - Android Tablets and Multiple User Accounts 2385
  - Determining What the User Can Do 2391
  - ***Impacts of Device-Level Restrictions*** ***2393***
  - Enabling Custom Restrictions 2394
  - Implicit Intents May Go "Boom" 2404
- Tapjacking
  - Prerequisites 2407
  - What is Tapjacking? 2407
  - Detecting Potential Tapjackers 2411
  - Defending Against Tapjackers 2414
  - Why Is This Being Discussed? 2417
  - What Changed in 4.0.3? 2418
- Miscellaneous Security Techniques
  - Prerequisites 2419
  - Public Key Validation 2419
  - Choosing Your Signing Keysize 2430
  - Avoiding Accidental APIs 2431
  - Other Ways to Expose Data 2436
- AlarmManager and the Scheduled Service Pattern
  - Scenarios 2439
  - Options 2440

- A Simple Example 2442
- The Five set...() Varieties 2444
- The Four Types of Alarms 2446
- When to Schedule Alarms 2446
- Archetype: Scheduled Service Polling 2448
- ***Staying Awake at Work 2451***
- Warning: Not All Android Devices Play Nice 2456
- Debugging Alarms 2456
- WakefulBroadcastReceiver 2459
- Android 6.0 and the War on Background Processing 2462

- PowerManager and WakeLocks
  - Prerequisites 2463
  - Keeping the Screen On, UI-Style 2463
  - The Role of the WakeLock 2464
  - What WakefulIntentService Does 2465
- JobScheduler
  - Prerequisites 2467
  - The Limitations of AlarmManager 2467
  - Enter the JobScheduler 2468
  - Employing JobScheduler 2468
  - Pondering Backoff Criteria 2480
  - Other JobScheduler Features 2482
  - ***Android 6.0 and "the War on Background Processing" 2483***
  - GCM Network Manager 2497
- Accessing Location-Based Services
  - Prerequisites 2499
  - Location Providers: They Know Where You're Hiding 2500
  - Finding Yourself 2500
  - On the Move 2502
  - Are We There Yet? Are We There Yet? Are We There Yet? 2503
  - Testing... Testing... 2504
  - Alternative Flavors of Updates 2505
  - The Fused Option 2506
- The Fused Location Provider
  - Prerequisites 2507
  - Why Use the Fused Location Provider? 2507
  - Why Not Use the Fused Location Provider? 2508
  - ***Finding Our Location, Once 2508***
  - Requesting Location Updates 2514
  - ***I Can Haz Location? 2516***
- Working with the Clipboard

- Prerequisites 2525
- Using the Clipboard on Android 1.x/2.x 2525
- Advanced Clipboard on Android 3.x and Higher 2528
- Monitoring the Clipboard 2533
- The Android 4.3 Clipboard Bug 2535

- Telephony
  - Prerequisites 2537
  - Report To The Manager 2538
  - You Make the Call! 2538
  - No, Really, You Make the Call! 2541
- Working With SMS
  - Prerequisites 2543
  - Sending Out an SOS, Give or Take a Letter 2544
  - Monitoring and Receiving SMS 2550
  - The SMS Inbox 2557
  - Asking to Change the Default 2558
  - SMS and the Emulator 2559
- NFC
  - Prerequisites 2561
  - What Is NFC? 2561
  - To NDEF, Or Not to NDEF 2563
  - NDEF Modalities 2563
  - NDEF Structure and Android's Translation 2564
  - The Reality of NDEF 2565
  - Sources of Tags 2567
  - Writing to a Tag 2567
  - Responding to a Tag 2575
  - Expected Pattern: Bootstrap 2576
  - Mobile Devices are Mobile 2577
  - Enabled and Disabled 2577
  - Android Beam 2577
  - Beaming Files 2584
  - Another Sample: SecretAgentMan 2586
  - Additional Resources 2594
- Device Administration
  - Prerequisites 2595
  - Objectives and Scope 2595
  - Defining and Registering an Admin Component 2596
  - Going Into Lockdown 2602
  - Passwords and Device Administration 2609
  - Getting Along with Others 2613

- Basic Use of Sensors
  - Prerequisites 2615
  - The Sensor Abstraction Model 2615
  - Considering Rates 2616
  - Reading Sensors 2617
  - Batching Sensor Readings 2627
- Printing and Document Generation
  - Prerequisites 2630
  - The Android Print System 2630
  - About the Sample App 2631
  - Printing a Bitmap 2632
  - Printing an HTML Document 2634
  - Printing a PDF File 2638
  - Printing Using a Canvas 2646
  - Print Jobs 2648
  - Printing, Threads, and Services 2648
  - Printing Prior to Android 4.4 2651
  - HTML Generation 2651
  - PDF Generation Options 2655
- Other System Settings and Services
  - Prerequisites 2657
  - Setting Expectations 2657
  - Can You Hear Me Now? OK, How About Now? 2662
  - The Rest of the Gang 2665
- Dealing with Different Hardware
  - Prerequisites 2667
  - ***Filtering Out Devices 2667***
  - ***Runtime Capability Detection 2670***
  - Dealing with Device Bugs 2671
- Writing and Using Parcelables
  - Prerequisites 2673
  - The Role of Parcelable 2673
  - Writing a Parcelable 2674
  - The Limitations of Parcelable 2680
- Responding to URLs
  - Prerequisites 2685
  - Manifest Modifications 2685
  - Creating a Custom URL 2687
  - Reacting to the Link 2688
  - App Links 2690
- Plugin Patterns

- Prerequisites 2699
- Definitions, Scenarios, and Scope 2699
- The Keys to Any Plugin System 2700
- Case Study: DashClock 2708
- Other Plugin Examples 2711

- PackageManager Tricks
  - Prerequisites 2729
  - Asking Around 2729
  - ***Preferred Activities 2733***
  - Middle Management 2737
- Searching with SearchManager
  - Prerequisites 2741
  - Hunting Season 2741
  - Search Yourself 2743
  - Searching for Meaning In Randomness 2749
  - May I Make a Suggestion? 2751
  - Putting Yourself (Almost) On Par with Google 2755
- Remote Services and the Binding Pattern
  - Prerequisites 2761
  - The Binding Pattern 2762
  - When IPC Attacks! 2768
  - Service From Afar 2770
  - ***Tightening Up the Security 2776***
  - Servicing the Service 2778
  - Thinking About Security 2783
  - The "Everlasting Service" Anti-Pattern 2783
- Advanced Manifest Tips
  - Prerequisites 2785
  - Just Looking For Some Elbow Room 2785
  - Using an Alias 2794
  - Getting Meta (Data) 2796
- Miscellaneous Integration Tips
  - Prerequisites 2801
  - Take the Shortcut 2801
  - Homing Beacons for Intents 2808
  - ***Integrating with Text Selection 2808***
- Reusable Components
  - Prerequisites 2821
  - Where Do I Find Them? 2821
  - How Are They Packaged? 2822
  - How Do I Create Them? 2823

- Other Considerations for Publishing Reusable Code 2826
- Android Studio Editors and Dialogs
  - Prerequisites 2829
  - ***Project Structure 2829***
  - Translations Editor 2838
- Advanced Emulator Capabilities
  - Prerequisites 2845
  - x86 Images 2845
  - Hardware Graphics Acceleration 2848
  - Defining New Devices 2851
  - Keyboard Behavior 2854
  - Headless Operation 2854
- Using Lint
  - Prerequisites 2855
  - What It Is 2855
  - When It Runs 2856
  - What to Fix 2859
  - What to Configure 2859
- Using Hierarchy View
  - Prerequisites 2865
  - Launching Hierarchy View 2865
  - Viewing the View Hierarchy 2866
  - ViewServer 2869
- Using DDMS
  - Prerequisites 2871
  - Starting DDMS 2871
  - File Push and Pull 2872
  - Screenshots 2873
  - Location Updates 2873
  - Placing Calls and Messages 2874
- Issues with Speed
  - Prerequisites 2877
  - Getting Things Done 2877
  - Your UI Seems... Janky 2878
  - Not Far Enough in the Background 2878
  - Playing with Speed 2879
- Finding CPU Bottlenecks
  - Prerequisites 2881
  - Traceview 2882
  - Other General CPU Measurement Techniques 2893
  - UI “Jank” Measurement 2894

- Focus On: NDK
  - Prerequisites 2911
  - The Role of the NDK 2912
  - NDK Installation and Project Setup 2915
  - Writing Your Makefile(s) 2919
  - Building Your Library 2920
  - Using Your Library Via JNI 2921
  - Building and Deploying Your Project 2926
  - ***Gradle and the NDK 2927***
- Improving CPU Performance in Java
  - Prerequisites 2935
  - Reduce CPU Utilization 2935
  - Reduce Time on the Main Application Thread 2940
  - Improve Throughput and Responsiveness 2948
- Finding and Eliminating Jank
  - Prerequisites 2951
  - The Case: ThreePaneDemoBC 2951
  - Are We Janky? 2952
  - Finding the Source of the Jank 2952
  - Where Things Went Wrong 2962
  - Removing the Jank 2963
- Issues with Bandwidth
  - Prerequisites 2965
  - You're Using Too Much of the Slow Stuff 2966
  - You're Using Too Much of the Expensive Stuff 2966
  - You're Using Too Much of Somebody Else's Stuff 2967
  - You're Using Too Much... And There Is None 2968
- Focus On: TrafficStats
  - Prerequisites 2969
  - TrafficStats Basics 2969
  - Example: TrafficMonitor 2971
  - Other Ways to Employ TrafficStats 2979
- Measuring Bandwidth Consumption
  - Prerequisites 2981
  - On-Device Measurement 2981
  - Off-Device Measurement 2983
  - Tactical Measurement in DDMS 2985
- Being Smarter About Bandwidth
  - Prerequisites 2989
  - Bandwidth Savings 2989
  - ***Bandwidth Shaping 2994***

- Avoiding Metered Connections 2998
- Issues with Application Heap
  - Prerequisites 2999
  - You Are in a Heap of Trouble 3000
  - Determining Your Heap Size At Runtime 3000
  - Fragments of Memory 3001
  - Getting a Trim 3002
  - Warning: Contains Graphic Images 3003
  - Releasing SQLite Memory 3014
  - In Too Deep (on the Stack) 3014
- Finding Memory Leaks with MAT
  - Prerequisites 3017
  - Setting Up MAT 3017
  - Getting Heap Dumps 3018
  - Basic MAT Operation 3024
  - Some Leaks and Their MAT Analysis 3031
- Issues with System RAM
  - Prerequisites 3039
  - ***Can't We All Just Get Along? 3039***
  - Contributors to System RAM Consumption 3040
  - Measuring System RAM Consumption: Tools 3041
  - Measuring System RAM Consumption: Runtime 3056
  - Learn To Let Go (Of Your Heap) 3057
- Issues with Battery Life
  - Prerequisites 3059
  - You're Getting Blamed 3060
  - Not All Batteries Are Created Equal 3061
  - Stretching Out the Last mWh 3061
- Power Measurement Options
  - Prerequisites 3063
  - batterystats and the Battery Historian 3064
  - The Qualcomm Tool (That Must Not Be Named) 3074
  - PowerTutor 3075
  - Battery Screen in Settings Application 3079
  - BatteryInfo Dump 3081
- Sources of Power Drain
  - Prerequisites 3085
  - Screen 3086
  - Disk I/O 3087
  - ***WiFi and Mobile Data 3088***
  - GPS 3091

- Camera 3092
- Additional Sources 3092

- Addressing Application Size Issues
  - Prerequisites 3095
  - ***Java Code, and the 64K Method Limit 3095***
  - Native Code 3099
  - Images 3101
  - APK Expansion Files 3103
- The Role of Scripting Languages
  - Prerequisites 3105
  - All Grown Up 3105
  - Following the Script 3106
  - Going Off-Script 3107
- The Scripting Layer for Android
  - Prerequisites 3111
  - The Role of SL4A 3111
  - Getting Started with SL4A 3112
  - Writing SL4A Scripts 3120
  - Running SL4A Scripts 3125
  - Potential Issues 3128
- JVM Scripting Languages
  - Prerequisites 3131
  - Languages on Languages 3131
  - A Brief History of JVM Scripting 3132
  - Limitations 3133
  - SL4A and JVM Languages 3134
  - Embedding JVM Languages 3134
  - Other JVM Scripting Languages 3147
- The Role of Alternative Environments
  - Prerequisites 3151
  - In the Beginning, There Was Java... 3152
  - ... And It Was OK 3152
  - Bucking the Trend 3153
  - Support, Structure 3153
  - Caveat Developer 3154
- HTML5
  - Prerequisites 3155
  - Offline Applications 3155
  - Web Storage 3162
  - Going To Production 3165
  - Issues You May Encounter 3166

- HTML5: The Baseline 3169
- PhoneGap
  - Prerequisites 3171
  - What Is PhoneGap? 3171
  - Using PhoneGap 3174
  - PhoneGap and the Checklist Sample 3179
  - Issues You May Encounter 3184
  - For More Information 3187
- Other Alternative Environments
  - Prerequisites 3189
  - Rhodes 3189
  - Flash, Flex, and AIR 3190
  - JRuby and Ruboto 3190
  - App Inventor 3191
  - Titanium Mobile 3193
  - Other JVM Compiled Languages 3194
- In-App Diagnostics
  - ***Prerequisites 3197***
  - The Diagnostic Activity 3198
  - ***The Diagnostic Web App 3207***
  - ***The Diagnostic Overlay 3216***
- Anti-Patterns
  - Prerequisites 3231
  - Leak Threads... Or Things Attached to Threads 3231
  - Use Large Heap Unnecessarily 3233
  - Misuse the MENU Button 3235
  - Interfere with Navigation 3236
  - Use android:sharedUserId 3238
  - Implement a “Quit” Button 3239
  - Terminate Your Process 3241
  - Try to Hide from the User 3242
  - Use Multiple Processes 3243
  - Hog System Resources 3245
- Widget Catalog: AdapterViewFlipper
  - ***Key Usage Tips 3247***
  - A Sample Usage 3248
  - Visual Representation 3248
- Widget Catalog: CalendarView
  - Key Usage Tips 3249
  - A Sample Usage 3250
  - ***Visual Representation 3250***

- Widget Catalog: DatePicker
  - ***Key Usage Tips*** ***3253***
  - A Sample Usage 3254
  - Visual Representation 3255
- Widget Catalog: ExpandableListView
  - Key Usage Tips 3261
  - A Sample Usage 3262
  - Visual Representation 3267
- Widget Catalog: SeekBar
  - Key Usage Tips 3271
  - A Sample Usage 3271
  - Visual Representation 3273
- Widget Catalog: SlidingDrawer
  - Key Usage Tips 3275
  - A Sample Usage 3276
  - Visual Representation 3277
- Widget Catalog: StackView
  - Key Usage Tips 3279
  - A Sample Usage 3280
  - Visual Representation 3281
- Widget Catalog: TabHost and TabWidget
  - Deprecation Notes 3283
  - Key Usage Tips 3283
  - A Sample Usage 3284
  - Visual Representation 3286
- Widget Catalog: TimePicker
  - Key Usage Tips 3289
  - A Sample Usage 3289
  - ***Visual Representation*** ***3291***
- Widget Catalog: ViewFlipper
  - Key Usage Tips 3295
  - A Sample Usage 3296
  - Visual Representation 3297
- Chrome and Chrome OS
  - Prerequisites 3300
  - Welcome to the ARC 3300
  - The ARC Welder 3300
  - ***Packaging Your Android App for Chrome*** ***3302***
  - Distribution Options 3307
  - ***Trying Your App*** ***3310***
  - ***Your App on Chrome OS*** ***3311***

- Apps Sans Role 3322
- Other Security Notes 3322
- Getting Help 3322

- Device Catalog: Kindle Fire
  - Prerequisites 3325
  - Introducing the Kindle Fire series 3325
  - What Features and Configurations Does It Use? 3326
  - ***What Is Really Different?* 3328**
  - Getting Your Development Environment Established 3334
  - How Does Distribution Work? 3336
  - Amazon Equivalents of Google Services 3337
  - Getting Help with the Kindle Fire 3338
- Device Catalog: BlackBerry
  - I Thought BlackBerry Had Their Own OS? 3339
  - ***What Else Is Different?* 3340**
  - What Are We Making? 3342
  - Getting Your Development Environment Established 3343
  - How Does Distribution Work? 3345
- Device Catalog: Wrist Wearables
  - Prerequisites 3348
  - Divvying Up the Wearables Space 3348
  - Example Wrist Wearables 3349
  - Strategic Considerations 3352
  - Tactical Considerations 3354
  - What About Android Wear? 3357
- Device Catalog: Android TV
  - Prerequisites 3359
  - Hey, Wait a Minute... I Thought the Name Was “Google TV”? 3359
  - Some Android TV Hardware 3360
  - What Features and Configurations Does It Use? 3362
  - What Is Really Different? 3363
  - Getting Your Development Environment Established 3365
  - How Does Distribution Work? 3367
- Device Catalog: Amazon Fire TV and Fire TV Stick
  - Prerequisites 3371
  - Introducing the Fire TV Devices 3371
  - What Features and Configurations Do They Use? 3377
  - What Is Really Different? 3379
  - Casting and Fire TV 3380
  - Getting Your Development Environment Established 3381
  - Working with the Remote and Controller 3383

- How Does Distribution Work? 3385
- Getting Help 3385

- Appendix A: CWAC Libraries
  - cwac-adapter 3387
  - ***cwac-cam2 ..... 3387***
  - cwac-colormixer 3388
  - cwac-layouts 3388
  - cwac-merge 3388
  - cwac-pager 3389
  - cwac-presentation 3389
  - cwac-provider 3389
  - cwac-richedit 3389
  - cwac-sacklist 3389
  - cwac-security 3390
  - cwac-strictmodeex 3390
  - cwac-wakeful 3390
- Appendix B: Android 6.0 Developer Preview
  - ***Updating to Android 6.0 ..... 3391***
  - ***What's New in Android 6.0? 3395***

Preface

Welcome to the Book!

Thanks!

Thanks for your interest in developing applications for Android! Android has grown from nothing to arguably the world's most popular smartphone OS in a few short years. Whether you are developing applications for the public, for your business or organization, or are just experimenting on your own, I think you will find Android to be an exciting and challenging area for exploration.

And, most of all, thanks for your interest in this book! I sincerely hope you find it useful and at least occasionally entertaining.

The Book's Structure

As you may have noticed, this is a rather large book.

To make the equivalent of ~3,500+ pages of material manageable, the chapters are divided into the *core* chapters and a series of *trails*.

The core chapters represent many key concepts that Android developers need to understand in order to build an app. While an occasional "nice to have" topic will drift into the core — to help illustrate a point, for example — the core chapters generally are fairly essential.

The core chapters are designed to be read in sequence and will interleave both traditional technical book prose with tutorial chapters, to give you hands-on experience with the concepts being discussed. Most of the tutorials can be skipped,

though the first two — covering setting up your SDK environment and creating a project – everybody should read.

The bulk of the chapters are divided into trails, covering some particular general topic, from data storage to advanced UI effects to performance measurement and tuning. Each trail will have several chapters. However, those chapters, and the trails themselves, are not necessarily designed to be read in any order. Each chapter in the trails will point out prerequisite chapters or concepts that you will want to have covered in advance. Hence, these chapters are mostly reference material, for when you specifically want to learn something about a specific topic.

The core chapters will link to chapters in the trails, to show you where you can find material related to the chapter you just read. So between the book's table of contents, this preface, the search tool in your digital book reader, and the cross-chapter links, you should have plenty of ways of finding the material you want to read.

You are welcome to read the entire book front-to-back if you wish. The trails will appear after the core chapters. Those trails will be in a reasonably logical order, though you may have to hop around a bit to cover all of the prerequisites.

The Trails

Here is a list of all of the trails and the chapters that pertain to those trails, in order of appearance (except for those appearing in the list multiple times, where they span major categories):

Code Organization and Gradle

- Working with Library Projects
- Gradle and Legacy Projects
- Gradle and Tasks
- Gradle and the New Project Structure
- Gradle and Dependencies
- Manifest Merger Rules
- Signing Your App
- Distribution
- Advanced Gradle for Android Tips

Testing

- JUnit and Android
- Testing with JUnit4
- MonkeyRunner and the Test Monkey
- Testing with UIAutomator

Advanced UI

- Notifications
- Advanced Notifications
- Introducing GridLayout
- Dialogs and DialogFragments
- Advanced ListViews
- Action Bar Navigation
- Action Modes and Context Menus
- Other Advanced Action Bar Techniques
- Toolbar
- AppCompat: The Official Action Bar Backport
- ActionBarSherlock
- RecyclerView
- Implementing a Navigation Drawer
- The Android Design Support Library
- Advanced Uses of WebView
- The Input Method Framework
- Fonts
- Rich Text
- Animators
- Legacy Animations
- Custom Drawables
- Mapping with Maps V2
- Crafting Your Own Views
- Advanced Preferences
- Custom Dialogs and Preferences
- Progress Indicators
- More Fun with Pagers
- Focus Management and Accessibility
- Miscellaneous UI Tricks
- Event Bus Alternatives
- Tasks

- The Assist API ("Now On Tap")

Home Screen Effects

- Home Screen App Widgets
- Adapter-Based App Widgets

Data Storage and Retrieval

- Content Provider Theory
- Content Provider Implementation Patterns
- The Loader Framework
- The ContactsContract Provider
- The CalendarContract Provider
- The MediaStore Provider
- Consuming Documents
- Providing Documents
- Encrypted Storage
- Packaging and Distributing Data
- Advanced Database Techniques

Advanced Network Topics

- SSL
- Embedding a Web Server
- Miscellaneous Network Capabilities

Media

- Audio Playback
- Audio Recording
- Video Playback
- Using the Camera via 3rd-Party Apps
- Working Directly with the Camera
- The MediaStore Provider
- Media Routes
- Supporting External Displays
- Google Cast and ChromeCast
- The "10 Foot UI"
- Putting the TVs All Together: Decktastic

- Creating a MediaRouteProvider
- Screenshots and Screen Recordings

Security

- SSL
- Encrypted Storage
- Advanced Permissions
- Restricted Profiles and UserManager
- Tapjacking
- Miscellaneous Security Techniques

Hardware and System Services

- AlarmManager and the Scheduled Service Pattern
- PowerManager and WakeLocks
- JobScheduler
- Accessing Location-Based Services
- The Fused Location Provider
- Working with the Clipboard
- Telephony
- Working With SMS
- NFC
- Device Administration
- Basic Use of Sensors
- Printing and Document Generation
- Other System Settings and Services
- Dealing with Different Hardware

Integration and Introspection

- Writing and Using Parcelables
- Responding to URLs
- Plugin Patterns
- PackageManager Tricks
- Searching with SearchManager
- Remote Services and the Binding Pattern
- Advanced Manifest Tips
- Miscellaneous Integration Tips
- Reusable Components

Other Tools

- Android Studio Dialogs and Editors
- Advanced Emulator Capabilities
- Using Lint
- Using Hierarchy View
- Using DDMS
- Finding CPU Bottlenecks
- Finding Memory Leaks with MAT

Tuning Android Applications

- Issues with Speed
- Finding CPU Bottlenecks
- NDK
- Improving CPU Performance in Java
- Finding and Eliminating Jank
- Issues with Bandwidth
- Focus On: TrafficStats
- Measuring Bandwidth Consumption
- Being Smarter About Bandwidth
- Issues with Application Heap
- Finding Memory Leaks with MAT
- Issues with System RAM
- Issues with Battery Life
- Other Power Measurement Options
- Sources of Power Drain
- Addressing Application Size Issues

Scripting Languages

- The Role of Scripting Languages
- The Scripting Layer for Android
- JVM Scripting Languages

Alternatives for App Development

- The Role of Alternative Environments
- HTML5
- PhoneGap

- Other Alternative Environments

Miscellaneous Topics

- In-App Diagnostics
- Anti-Patterns

Widget Catalog

- AdapterViewFlipper
- CalendarView
- DatePicker
- ExpandableListView
- SeekBar
- SlidingDrawer
- StackView
- TabHost
- TimePicker
- ViewFlipper

Device Catalog

- Chrome and Chrome OS
- Kindle Fire
- BlackBerry
- Wrist Wearables
- Google TV
- Amazon Fire TV

Appendices

- Appendix A: CWAC Libraries
- Appendix B: Android 6.0 Developer Preview

About the Updates

This book is updated frequently, typically every 2-3 months.

Each release has notations to show what is new or changed compared with the immediately preceding release:

- The Table of Contents shows sections with changes in bold-italic font
- Those sections have changebars on the right to denote specific paragraphs that are new or modified

Warescription

You (hopefully) are reading this digital book by means of a Warescription.

The Warescription entitles you, for the duration of your subscription, to digital editions of this book and its updates, in PDF, EPUB, and Kindle (MOBI/KF8) formats. You also have access to a version of the book as its own Android APK file, complete with high-speed full-text searching. You also have access to other titles that CommonsWare may publish during that subscription period.

Each subscriber gets personalized editions of all editions of each title. That way, your books are never out of date for long, and you can take advantage of new material as it is made available. For example, when new releases of the Android SDK are made available, this book will be quickly updated to be accurate with changes in the APIs.

However, you can only download the books if either you have an active Warescription, or until the book is updated after your Warescription expires. Hence, **please download your updates as they come out**. You can find out when new releases of this book are available via:

1. The CommonsWare Twitter feed
2. The CommonsBlog
3. The Warescription newsletter, which you can subscribe to off of your Warescription page
4. Just check back on the Warescription site every month or two

Subscribers also have access to other benefits, including:

- "Office hours" — online chats to help you get answers to your Android application development questions. You will find a calendar for these on your Warescription page.
- A Stack Overflow "bump" service, to get additional attention for a question that you have posted there that does not have an adequate answer.

Book Bug Bounty

Find a problem in one of our books? Let us know!

Be the first to report a unique concrete problem in the current digital edition, and we will extend your Warescription by six months as a bounty for helping us deliver a better product.

By "concrete" problem, we mean things like:

1. Typographical errors
2. Sample applications that do not work as advertised, in the environment described in the book
3. Factual errors that cannot be open to interpretation

By "unique", we mean ones not yet reported. Be sure to check the book's errata page, though, to see if your issue has already been reported. One coupon is given per email containing valid bug reports.

We appreciate hearing about "softer" issues as well, such as:

1. Places where you think we are in error, but where we feel our interpretation is reasonable
2. Places where you think we could add sample applications, or expand upon the existing material
3. Samples that do not work due to "shifting sands" of the underlying environment (e.g., changed APIs with new releases of an SDK)

However, those "softer" issues do not qualify for the formal bounty program.

Questions about the bug bounty, or problems you wish to report for bounty consideration, should be sent to bounty@commonsware.com.

Source Code and Its License

The source code samples shown in this book are available for download from the book's GitHub repository. All of the Android projects are licensed under the Apache 2.0 License, in case you have the desire to reuse any of it.

If you wish to use the source code from the GitHub repository, please follow the instructions on that repository's home page for details of how to use the projects in various development environments, notably Android Studio.

Copying source code directly from the book, in the PDF editions, works best with Adobe Reader, though it may also work with other PDF viewers. Some PDF viewers, for reasons that remain unclear, foul up copying the source code to the clipboard when it is selected.

Creative Commons and the Four-to-Free (42F) Guarantee

Each CommonsWare book edition will be available for use under the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license as of the fourth anniversary of its publication date, or when 4,000 copies of the edition have been sold, whichever comes first. That means that, once four years have elapsed (perhaps sooner!), you can use this prose for non-commercial purposes. That is our Four-to-Free Guarantee to our readers and the broader community. For the purposes of this guarantee, new Warescriptions and renewals will be counted as sales of this edition, starting from the time the edition is published.

This edition of this book will be available under the aforementioned Creative Commons license on *1 October 2019*. Of course, watch the CommonsWare Web site, as this edition might be relicensed sooner based on sales.

For more details on the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license, visit the Creative Commons Web site

Note that future editions of this book will become free on later dates, each four years from the publication of that edition or based on sales of that specific edition. Releasing one edition under the Creative Commons license does not automatically release *all* editions under that license.

Acknowledgments

I would like to thank the Android team, not only for putting out a good product, but for invaluable assistance on the Android Google Groups and Stack Overflow.

I would also like to thank the thousands of readers of past editions of this book, for their feedback, bug reports, and overall support.

Of course, thanks are also out to the overall Android ecosystem, particularly those developers contributing their skills to publish libraries, write blog posts, answer support questions, and otherwise contribute to the strength of Android.

Core Chapters

Key Android Concepts

No doubt, you are in a hurry to get started with Android application development. After all, you are reading this book, aimed at busy coders.

However, before we dive into getting tools set up and starting in on actual programming, it is important that we "get on the same page" with respect to several high-level Android concepts. This will simplify further discussions later in the book.

Android Applications

This book is focused on writing Android applications. An application is something that a user might install from the Play Store or otherwise download to their device. That application should have some user interface, and it might have other code designed to work in the background (multi-tasking).

This book is not focused on modifications to the Android firmware, such as writing device drivers. For that, you will need to seek other resources.

This book assumes that you have some hands-on experience with Android devices, and therefore you are familiar with buttons like HOME and BACK, the built-in Settings application, the concept of a home screen and launcher, and so forth. If you have never used an Android device, you are strongly encouraged to get one (e.g., a used one on eBay, Craigslist, etc.) and spend some time with it before starting in on learning Android application development.

Programming Language

The vast majority of Android applications are written exclusively in Java. Hence, that is what this book will spend most of its time on and will demonstrate with a seemingly infinite number of examples.

However, there are other options:

- You can write parts of the app in C/C++, for performance gains, porting over existing code bases, etc.
- You can write an entire app in C/C++, mostly for games using OpenGL for 3D animations
- You can write the guts of an app in HTML, CSS, and JavaScript, using tools to package that material into an Android application that can be distributed through the Play Store and similar venues
- And so on

Coverage of these non-Java alternatives will be found in the trails of this book, as the bulk of this book is focused on Java.

The author assumes that you know Java at this point. If you do not, you will need to learn Java before you go much further. You do not need to know *everything* about Java, as Java is vast. Rather, focus on:

- Language fundamentals (flow control, etc.)
- Classes and objects
- Methods and data members
- Public, private, and protected
- Static and instance scope
- Exceptions
- Threads
- Collections
- Generics
- File I/O
- Reflection
- Interfaces

The links are to Wikibooks material on those topics, though there are countless other Java resources for you to consider.

Components

When you first learned Java — whether that was yesterday or back when dinosaurs roamed the Earth — you probably started off with something like this:

```
class SillyApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

In other words, the entry point into your application was a `public static void` method named `main()` that took a `String` array of arguments. From there, you were responsible for doing whatever was necessary.

However, there are other patterns used elsewhere in Java. For example, you do not usually write a `main()` method when writing a Java servlet. Instead, you extend a particular class supplied by a framework (e.g., `HttpServlet`) to create a component, then write some metadata that enumerates your components and tell the framework when and how to use them (e.g., `WEB.XML`).

Android apps are closer in spirit to the servlet approach. You will not write a `public static void main()` method. Instead, you will create subclasses of some Android-supplied base classes that define various application components. In addition, you will create some metadata that tells Android about those subclasses.

There are four types of components, all of which will be covered extensively in this book:

Activities

The building block of the user interface is the *activity*. You can think of an activity as being the Android analogue for the window or dialog in a desktop application, or the page in a classic Web app. It represents a chunk of your user interface and, in some cases, a discrete entry point into your app (i.e., a way for other apps to link to your app).

Normally, an activity will take up most of the screen, leaving space for some "chrome" bits like the clock, signal strength indicators, and so forth.

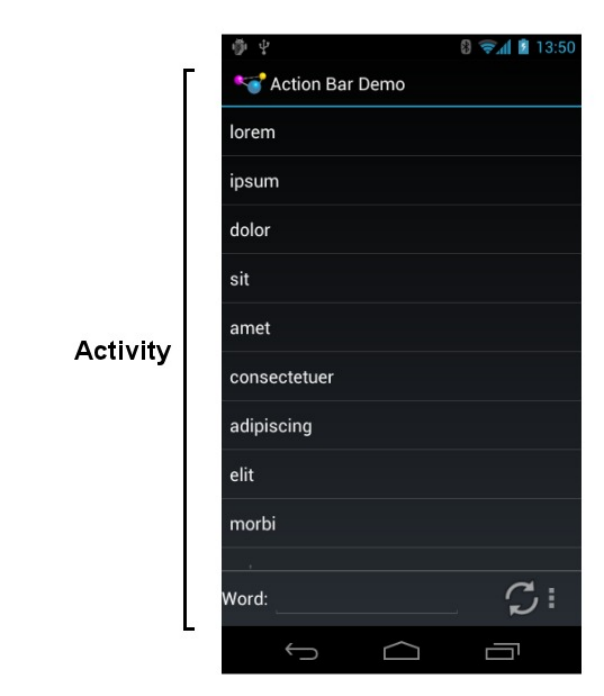


Figure 1: Activity on the screen

Services

Activities are short-lived and can be shut down at any time, such as when the user presses the BACK button. *Services*, on the other hand, are designed to keep running, if needed, independent of any activity, for a moderate period of time. You might use a service for checking for updates to an RSS feed, or to play back music even if the controlling activity is no longer operating. You will also use services for scheduled tasks (akin to Linux or OS X "cron jobs") and for exposing custom APIs to other applications on the device, though the latter is a relatively advanced capability.

Content Providers

Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications. The Android development model encourages you to make your own data available to other applications, as well as your own — building a content provider lets you do that, while maintaining a degree of control over how your data gets accessed.

Broadcast Receivers

The system, or applications, will send out *broadcasts* from time to time, for everything from the battery getting low, to when the screen turns off, to when connectivity changes from WiFi to mobile data. A broadcast receiver can arrange to listen for these broadcasts and respond accordingly.

Widgets, Containers, Resources, and Fragments

Most of the focus on Android application development is on the UI layer and activities. Most Android activities use what is known as "the widget framework" for rendering their user interface, though you are welcome to use the 2D (`Canvas`) and 3D (OpenGL) APIs as well for more specialized GUIs.

In Android terms, a *widget* is the "micro" unit of user interface. Fields, buttons, labels, lists, and so on are all widgets. Your activity's UI, therefore, is made up of one or more of these widgets. For example, here we see label (`TextView`), field (`EditText`), and push-button (`Button`) widgets:

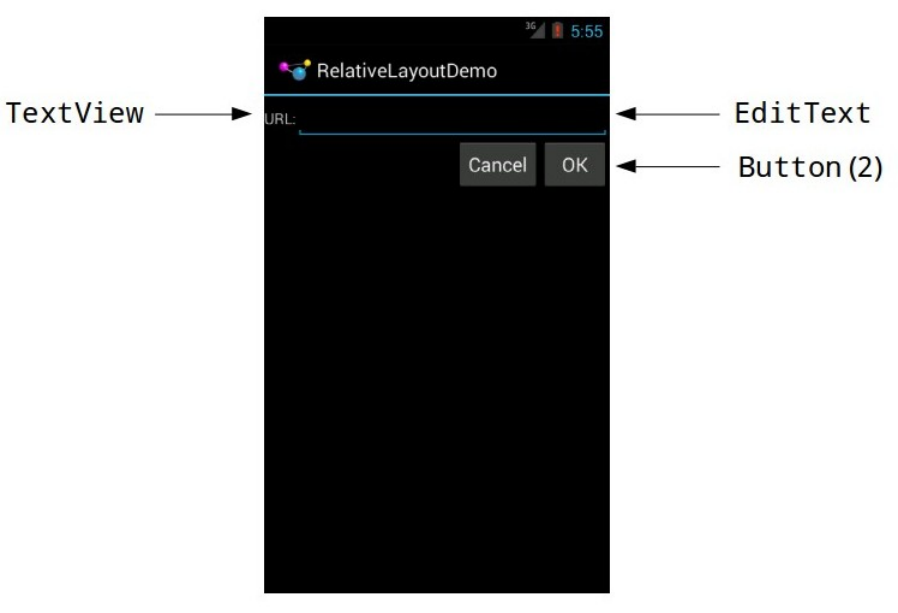


Figure 2: Activity with widgets

If you have more than one widget — which is fairly typical — you will need to tell Android how those widgets are organized on the screen. To do that, you will use

various container classes referred to as *layout managers*. These will let you put things in rows, columns, or more complex arrangements as needed.

To describe how the containers and widgets are connected, you will typically create a *layout resource file*. *Resources* in Android refer to things like images, strings, and other material that your application uses but is not in the form of some programming language source code. UI layouts are another type of resource. You will create these layouts either using a structured tool, such as an IDE's drag-and-drop GUI builder, or by hand in XML form.

Sometimes, your UI will work across all sorts of devices: phones, tablets, televisions, etc. Sometimes, your UI will need to be tailored for different environments. You will be able to put resources into *resource sets* that indicate under what circumstances those resources can be used (e.g., use these for normal-sized screens, but use those for larger screens).

We will be examining all of these concepts, in much greater detail, as we get deeper into the book.

Apps and Packages

Given a bucket of source code and a basket of resources, the Android build tools will give you an application as a result. The application comes in the form of an *APK file*. It is that APK file that you will upload to the Play Store or distribute by other means.

Each Android application has a package name, also referred to as an application ID. A package name must fulfill three requirements:

1. It must be a valid Java package name, as some Java source code will be generated by the Android build tools in this package
2. No two applications can exist on a device at the same time with the same application ID
3. No two applications can be uploaded to the Play Store having the same application ID

When you create your Android project — the repository of that source code and those resources — you will declare what package name is to be used for your app. Typically, you will pick a package name following the Java package name "reverse domain name" convention (e.g., `com.commonsware.android.foo`). That way, the domain name system ensures that your package name prefix (`com.commonsware`) is

unique, and it is up to you to ensure that the rest of the package name distinguishes one of your apps from any other.

Android Devices

There are well in excess of one billion Android devices in use today, representing thousands of different models from dozens of different manufacturers. Android itself has evolved since Android 1.0 in 2008. Between different device types and different Android versions, many a media pundit has lobbed the term "fragmentation" at Android, suggesting that creating apps that run on all these different environments is impossible.

In reality, it is not that bad. Some apps will have substantial trouble, but most apps will work just fine if you follow the guidance presented in this book and in other resources.

Types

Android devices come in all shapes, sizes, and colors. However, there are four dominant "form factors":

- the phone
- the tablet
- the television (TV)
- the wearable (smart watches, Google Glass, etc.)

You will often hear developers and pundits refer to these form factors, and this book will do so from time to time as well. However, it is important that you understand that Android has no built-in concept of a device being a "phone" or a "tablet" or a "TV". Rather, Android distinguishes devices based on capabilities and features. So, you will not see an `isPhone()` method anywhere, though you can ask Android:

- what is the screen size?
- does the device have telephony capability?
- etc.

Similarly, as you build your applications, rather than thinking of those four form factors, focus on what capabilities and features you need. Not only will this help you line up better with how Android wants you to build your apps, but it will make it easier for you to adapt to other form factors that will come about such as:

- airplane seat-back entertainment centers
- in-car navigation and entertainment devices
- and so on

The Emulator

While there are over a billion Android devices representing thousands of models, probably you do not have one of each model. You may only have a single piece of Android hardware. And if you do not even have that, you most certainly will want to acquire one before trying to publish an Android app.

To help fill in the gaps between the devices you have and the devices that are possible, the Android developer tools ship an *emulator*. The emulator behaves like a piece of Android hardware, but it is a program you run on your development machine. You can use this emulator to emulate many different devices, with different screen sizes and Android OS versions, by creating one or more Android virtual devices, or *AVDs*.

In an upcoming chapter, we will discuss how you install the Android developer tools and how you will be able to create these AVDs and run the emulator.

OS Versions and API Levels

Android has come a long way since the early beta releases from late 2007. Each new Android OS version adds more capabilities to the platform and more things that developers can do to exploit those capabilities.

Moreover, the core Android development team tries very hard to ensure forwards and backwards compatibility. An app you write today should work unchanged on future versions of Android (forwards compatibility), albeit perhaps missing some features or working in some sort of "compatibility mode". And there are well-trod paths for how to create apps that will work both on the latest and on previous versions of Android (backwards compatibility).

To help us keep track of all the different OS versions that matter to us as developers, Android has *API levels*. A new API level is defined when an Android version ships that contains changes that affect developers. When you create an emulator AVD to test your app, you will indicate what API level that emulator should emulate. When you distribute your app, you will indicate the oldest API level your app supports, so the app is not installed on older devices.

At the time of this writing, the API levels of significance to most Android developers are:

- API Level 10 (Android 2.3.3)
- API Level 16 (Android 4.1)
- API Level 17 (Android 4.2)
- API Level 19 (Android 4.4)
- API Level 21 (Android 5.0)
- API Level 22 (Android 5.1)

Here, "of significance" refers to API levels that have a reasonable number of Android devices — 5% or more, as reported by the "Platform Versions" dashboard chart.

The latest production API level for most form factors is 22, representing Android 5.1. API Level 23 is for Android 6.0, which at the time of this writing is only available as a developer preview edition. Android 6.0 should be rolling out to ordinary production hardware later in 2015.

Note that API Level 20 was used for the version of Android 4.4 running on the first-generation Android Wear devices. Unless you are specifically developing apps for Wear, you will not be worrying much about API Level 20.

Also, right now, we have the M Developer Preview. This is a preview of the next major version of Android, presumably to be numbered 5.2 or 6.0. That version should ship later in 2015. In the meantime, experienced developers can start playing with the preview and experimenting with the new APIs and capabilities. However, while Android development is strange in general, development using preview editions is even more strange. Hence, newcomers to Android are advised to avoid developer previews.

Dalvik

In terms of Android, Dalvik is a virtual machine (VM). Virtual machines are used by many programming languages, such as Java, Perl, and Smalltalk. The Dalvik VM is designed to work much like a Java VM, but optimized for embedded Linux environments.

So, what really goes on when somebody writes an Android application is:

1. Developers write Java-syntax source code, leveraging class libraries published by the Android project and third parties.

2. Developers compile the source code into Java VM bytecode, using the `javac` compiler that comes with the Java SDK.
3. Developers translate the Java VM bytecode into Dalvik VM bytecode, which is packaged with other files into a ZIP archive with the `.apk` extension (the APK file).
4. An Android device or emulator runs the APK file, causing the bytecode to be executed by an instance of a Dalvik VM.

From your standpoint, most of this is hidden by the build tools. You pour Java source code into the top, and the APK file comes out the bottom.

However, there will be places from time to time where the differences between the Dalvik VM and the traditional Java VM will affect application developers, and this book will point out some of them where relevant.

Note that Android is moving to a new runtime environment, called ART. However, the "Dalvik" term will still be used for the bytecode that is generated as part of building an APK.

Processes and Threads

When your application runs, it will do so in its own process. This is not significantly different than any other traditional operating system. Part of Dalvik's magic is making it possible for many processes to be running many Android applications at one time without consuming ridiculous amounts of RAM.

Android will also set up a batch of threads for running your app. The thread that your code will be executed upon, most of the time, is variously called the "main application thread" or the "UI thread". You do not have to set it up, but, as we will see later in the book, you will need to pay attention to what you do and do not do on that thread. You are welcome to fork your own threads to do work, and that is fairly common, though in some places Android handles that for you behind the scenes.

Don't Be Scared

Yes, this chapter threw a lot of terms at you. We will be going into greater detail on all of them in this book. However, Android is like a jigsaw puzzle with lots of interlocking pieces. To be able to describe one concept in detail, we will need to at least reference some of the others. Hence, this chapter was meant to expose you to terms, in hopes that they will sound vaguely familiar as we dive into the details.

Choosing Your Development Toolchain

Before you go much further in your Android endeavors (or, possibly, endeavours, depending upon your preferred spelling), you will need to determine what toolchain you will use to build your Android applications.

Android Studio

The next-generation Google-backed Android IDE is Android Studio. Based off of IntelliJ IDEA, Android Studio is the new foundation of Google's efforts to give Android developers top-notch development tools. While it only reached a version 1.0 status in December 2014, Android Studio had been in use for ~18 months prior to that in various early-access and beta stages. While it still has bugs, it is certainly stable enough for app development.

The next chapter contains a section with instructions on how to set up Android Studio.

Eclipse

Eclipse is also a popular IDE, particularly for Java development. Eclipse was Google's original IDE for Android development, by means of the Android Developer Tools (ADT) add-in, which gives the core of Eclipse awareness of Android. The ADT add-in, in essence, takes regular Eclipse operations and extends them to work with Android projects.

Note, though, that Google has discontinued maintenance of ADT. The Eclipse Foundation is setting up the "Andmore" project to try to continue work on allowing Eclipse to build Android apps. This book does not cover the Andmore project at this time, and developers are **strongly** encouraged to not use the ADT-enabled Eclipse from Google.

IntelliJ IDEA

While Android Studio is based on IntelliJ IDEA, you can still use the original IntelliJ IDEA for Android app development. A large subset of the Android Studio capabilities are available in the Android plugin for IDEA. Plus, the commercial IDEA Ultimate Edition will go beyond Android Studio in many areas outside of Android development.

Command-Line Builds via Gradle for Android

And, of course, you do not need to use an IDE at all. While this may sound sacrilegious to some, IDEs are not the only way to build applications. Much of what is accomplished via the ADT can be accomplished through command-line equivalents, meaning a shell and an editor is all you truly need. For example, the author of this book did not use an IDE for Android development until 2011.

The recommended way to build Android apps outside of an IDE is by means of Gradle. Google has published a Gradle plugin that teaches Gradle how to build Android apps. Android Studio itself uses Gradle for its builds, so a single build configuration (e.g., `build.gradle` files) can be used both from an IDE and from a build automation tool like a continuous integration server.

An upcoming chapter gets into more about what Gradle (and the Android Plugin for Gradle) are all about.

Yet Other Alternatives

Other IDEs have their equivalents of the ADT, albeit with minimal assistance from Google. For example, NetBeans has support via the NBAndroid add-on, and reportedly this has advanced substantially in the past few years.

You will also hear reference to using Apache Ant for doing command-line builds of Android apps. This has largely been supplanted by Gradle for Android at this time,

and support for Apache Ant will end soon. Newcomers to Android are encouraged to not invest time in new work with Apache Ant for Android development projects.

IDEs… And This Book

You are welcome to use Android Studio or Eclipse as you work through this book. You are welcome to use another IDE if you wish. You are even welcome to skip the IDE outright and just use an editor.

This book is focused primarily on demonstrating Android capabilities and the APIs for exploiting those capabilities. Hence, the sample code will work with any IDE. However, this book will cover some Android Studio- and Eclipse-specific instructions, since they are the predominant answers today.

The tutorials will have instructions for both Android Studio and Eclipse.

What We Are Not Covering

In the beginning (a.k.a., 2007), we were lucky to have any means of creating an Android app.

Nowadays, there seems to be no end to the means by which we can create an Android app.

There are a few of these "means", though, that are specifically out of scope for this book.

App Inventor

You may also have heard of a tool named App Inventor and wonder where it fits in with all of this.

App Inventor was originally created by an education group within Google, as a means of teaching students how to think about programming constructs (branches, loops, etc.) and create interesting output (Android apps) without classic programming in Java or other syntax-based languages. App Inventor is purely drag-and-drop, both of widgets *and application logic*, the latter by means of "blocks" that snap together to form logic chains.

App Inventor was donated by Google to MIT, which has recently re-opened it to the public.

However, App Inventor is a closed system — at the present time, it does not somehow generate Java code that you can later augment. That limits you to whatever App Inventor is natively capable of doing, which, while impressive in its own right, offers a small portion of the total Android SDK capabilities.

App Generators

There are a seemingly infinite number of "app generators" available as online services. These are designed mostly for creating apps for specific vertical markets, such as apps for restaurants or apps for grocers. The resulting apps are mostly "brochure-ware", with few capabilities beyond a mobile Web site, yet still requiring the user to find, download, and install the app. Few of these generators provide the source code to the generated app, to allow the apps to be customized beyond what the generator generates.

Tutorial #1 - Installing the Tools

Now, let us get you set up with the pieces and parts necessary to build an Android app.

NOTE: The instructions presented here are accurate as of the time of this writing. However, the tools change rapidly, and so these instructions may be out of date by the time you read this. Please refer to the Android Developers Web site for current instructions, using this as a base guideline of what to expect.

Step #1 - Checking Your Hardware Requirements

Compiling and building an Android application, on its own, is not especially hardware-intensive, except for very large projects. However, there are two commonly-used tools that demand more from your development machine: your IDE and the Android emulator. Of the two, the emulator poses the bigger problem.

The more RAM you have, the better. 8GB or higher is a very good idea if you intend to use an IDE and the emulator together.

A faster CPU is also a good idea. However, the Android emulator only utilizes a single core from your development machine. Hence, it is the single-core speed that matters. The best CPU to use is one that can leverage multiple cores to give what amounts to a faster single core, such as Intel's Core i7 with Turbo Boost. For an emulator simulating a larger-screened device (e.g., tablet, television), a Core i7 that can "boost" up to 3.4GHz makes development much more pleasant. Conversely, a CPU like a Core 2 Duo with a 2.5GHz clock speed results in a tablet emulator that is nearly unusable.

Step #2 - Setting Up Java and 32-Bit Linux Support

When you write Android applications, you typically write them in Java source code. That Java source code is then turned into the stuff that Android actually runs (Dalvik bytecode in an APK file).

You need to obtain and install the official Sun/Oracle Java SE SDK (JDK). You can obtain this from the Oracle Java Web site for Windows, OS X, and Linux. The plain JDK (sans any "bundles") should suffice. Follow the instructions supplied by Oracle or Apple for installing it on your machine. At the time of this writing, Android supports Java 6 and Java 7, though Java 7 is required for certain scenarios and therefore is recommended. Java 8 works, though you may have to do additional work to configure your IDE to have Java 8 emit Java 7-compatible bytecode.

Android also supports the OpenJDK, particularly on Linux environments.

What Android does *not* support are any other Java compilers, including the GNU Compiler for Java (GCJ).

If your development OS is Linux, make sure that you can run 32-bit Linux binaries. This may or may not already be enabled in your Linux distro. For example, on Ubuntu 14.10, you may need to run the following to get the 32-bit binary support installed that is needed by the Android build tools:

```
sudo apt-get install lib32z1 lib32ncurses5 lib32stdc++6
```

You may also need `lib32bz2-1.0`, depending on your version of Linux.

Step #3 - Install the Developer Tools

As noted in the previous chapter, there are a few developer tools that you can choose from.

This book's tutorials focus on Android Studio. You are welcome to attempt to use Eclipse, another IDE, or no IDE at all for building Android apps. However, you will need to translate some of the tutorials' IDE-specific instructions to be whatever is needed for your development toolchain of choice.

With that in mind, visit the Android Studio download page, download the ZIP file for your platform, and unZIP it to some likely spot on your hard drive. Windows users who choose to download the self-installing EXE can just run that file.

Android Studio can then be run from the `studio` batch file or shell script from your Android Studio installation's `bin/` directory.

At the time of this writing, the current production version of Android Studio is 1.4.x, and this book covers that version. If you are reading this in the future, you may be on a newer version of Android Studio, and there may be some differences between what you have and what is presented here.

Step #4 - Install the SDKs and Add-Ons

Next, we need to review what pieces of the Android SDK we have already and perhaps install some new items. To do that, you need to access the SDK Manager.

When you first run Android Studio, you may be asked if you want to import settings from some other prior installation of Android Studio:

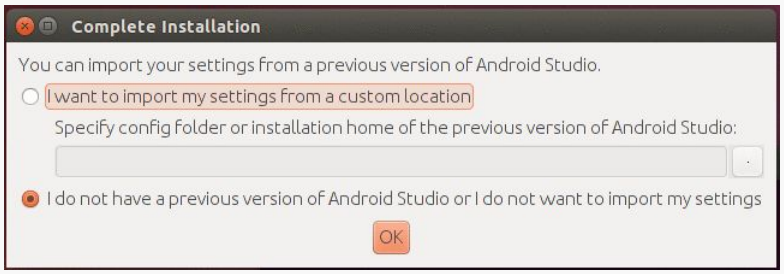


Figure 3: Android Studio First-Run Settings Migration Dialog

For most users, particularly those using Android Studio for the first time, the "I do not have..." option is the correct choice to make.

Then, after a short splash screen, you will be taken to the Android Studio Setup Wizard:

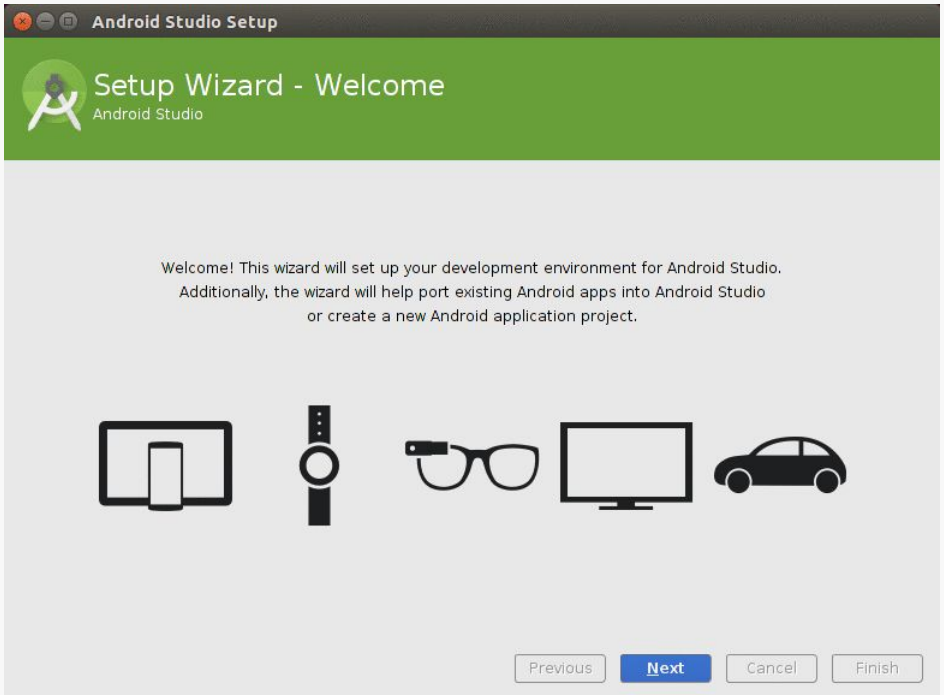


Figure 4: Android Studio Setup Wizard, First Page

Just click "Next" to advance to the second page of the wizard:

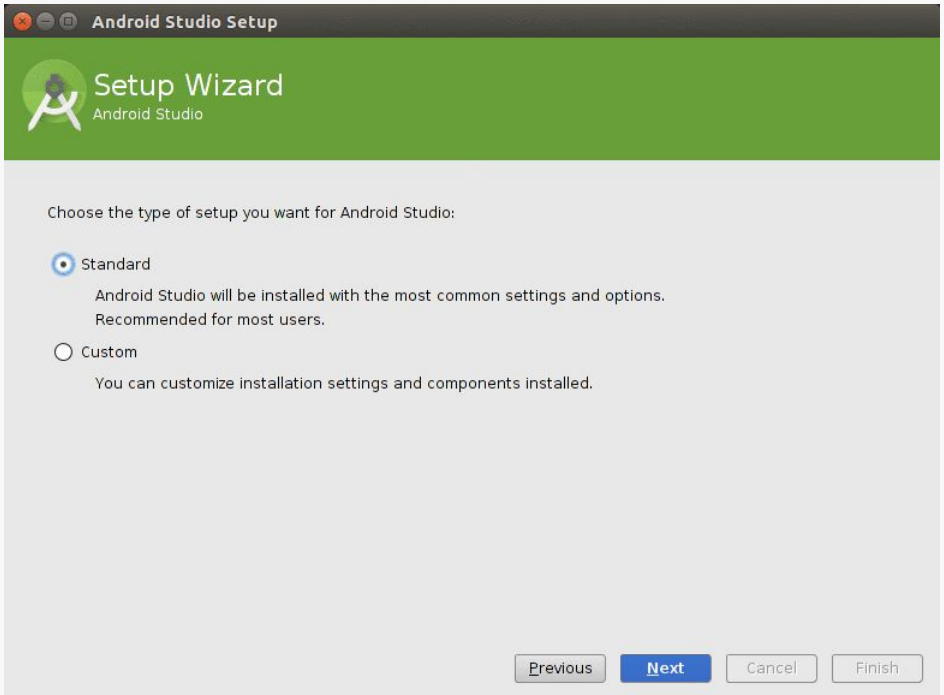


Figure 5: Android Studio Setup Wizard, Second Page

Here, you have a choice between "Standard" and "Custom" setup modes. Most likely, right now, the "Standard" route will be fine for your environment.

If you go the "Standard" route and click "Next", you may be taken to a wizard page explaining some information about the Android emulator:

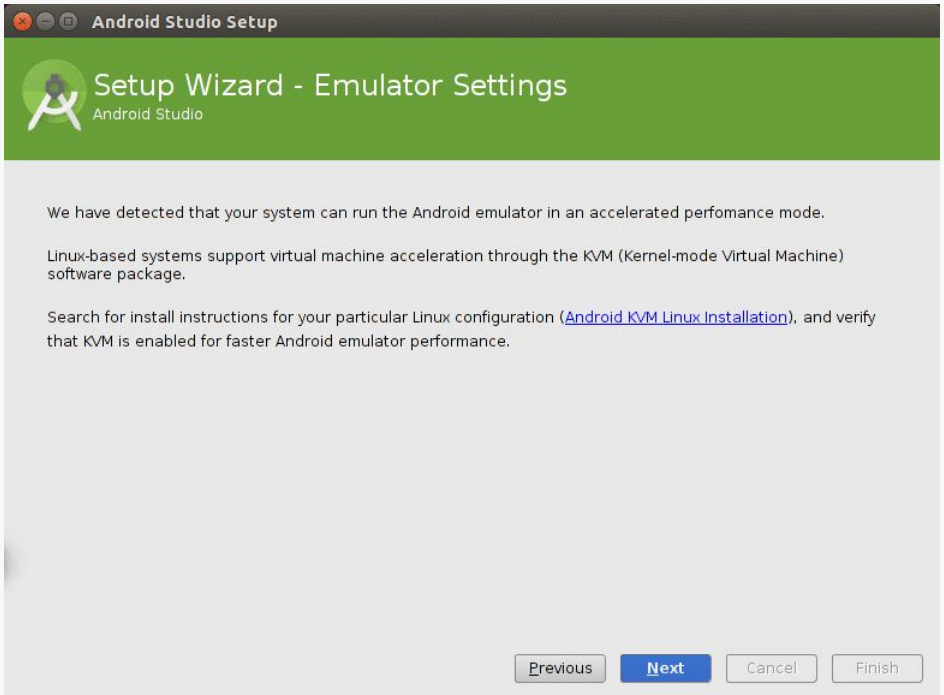


Figure 6: Android Studio Setup Wizard, Emulator Info Page

What is explained on this page may not make much sense to you. That is perfectly normal, and we will get into what this page is trying to say later in the book. Just click "Next" to advance to the next page where you will can review the roster of stuff that will be downloaded and installed for you:

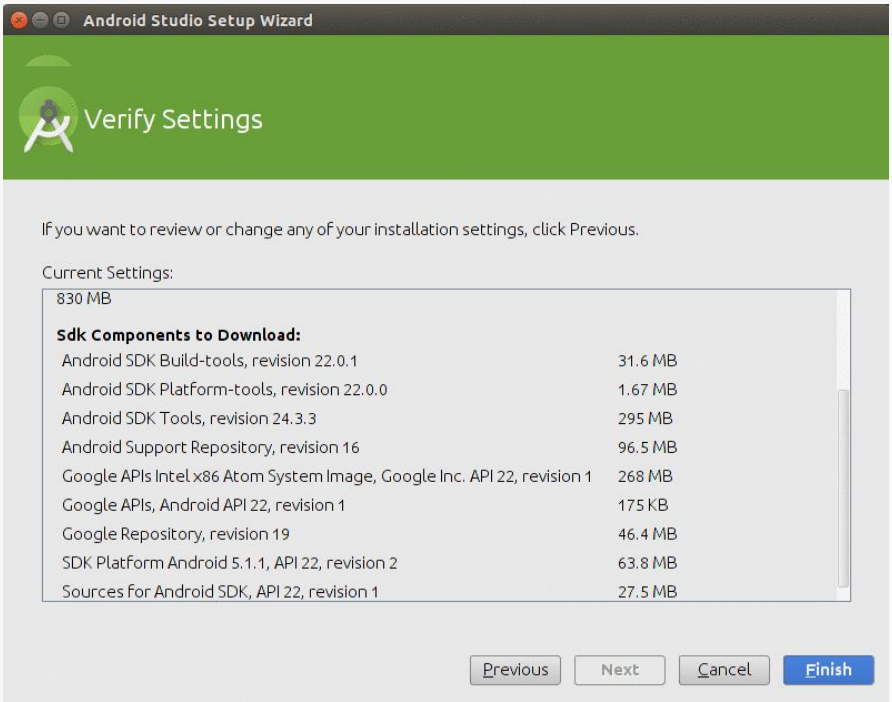


Figure 7: Android Studio Setup Wizard, Verify Settings Page

At this point, clicking "Finish" will begin the setup process. This will include downloading a copy of the Android SDK and installing it into a directory adjacent to where Android Studio itself is installed.

If you are running Linux, and your installation crashes with an "Unable to run mksdcard SDK tool" error, go back to Step #2 and set up 32-bit support on your Linux environment.

When that is done, after clicking "Finish" one last time, you will be taken to the Android Studio Welcome dialog:

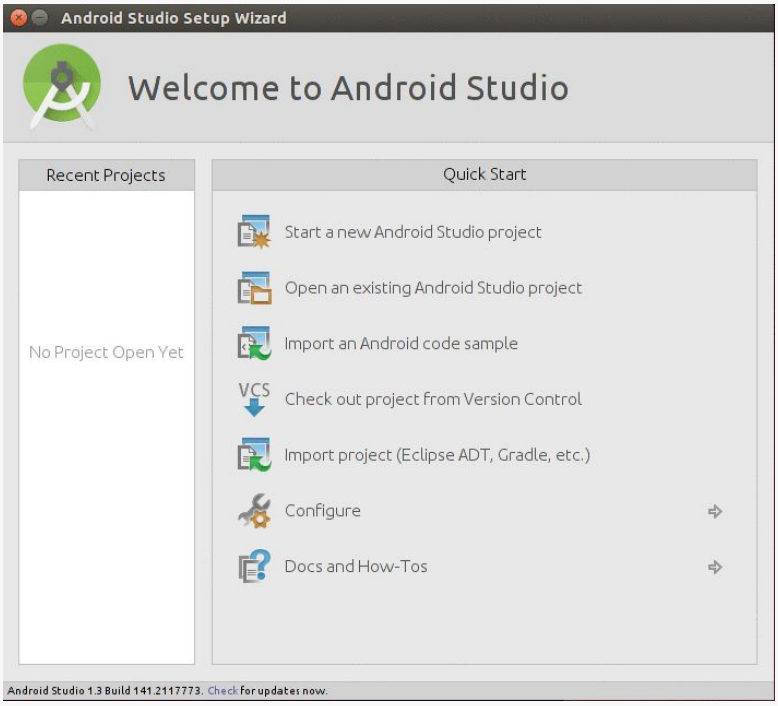


Figure 8: Android Studio Welcome Dialog

In your case, the contents of the "Recent Projects" list will be empty, as you have not created or opened any projects yet.

In very tiny print at the bottom of that dialog is a "Check for updates now" link. Click that, and if there are updates available, install them. This will automatically restart Android Studio. Android Studio *should* have downloaded the latest updates as part of the initial setup, so most likely this will indicate that nothing more is needed.

Then, in the welcome dialog, click Configure, to bring up a configuration sub-menu:

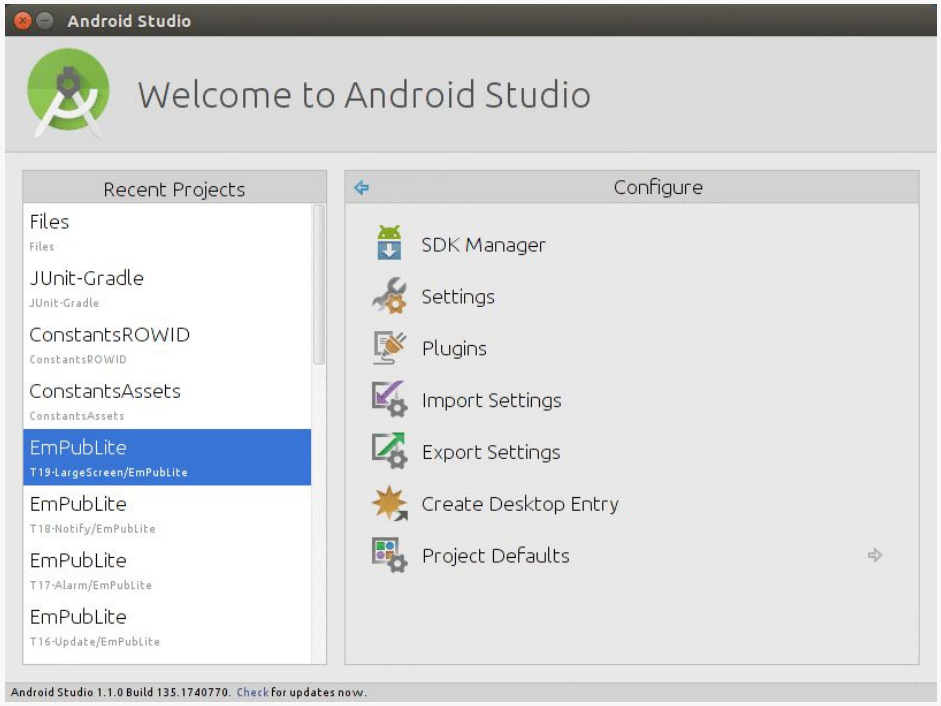


Figure 9: Android Studio Welcome Dialog, Configure Sub-Menu

There, tap on SDK Manager to bring up the SDK Manager.

Using SDK Manager and Updating Your Environment

You should now have the SDK Manager open, as part of the overall default settings for Android Studio:

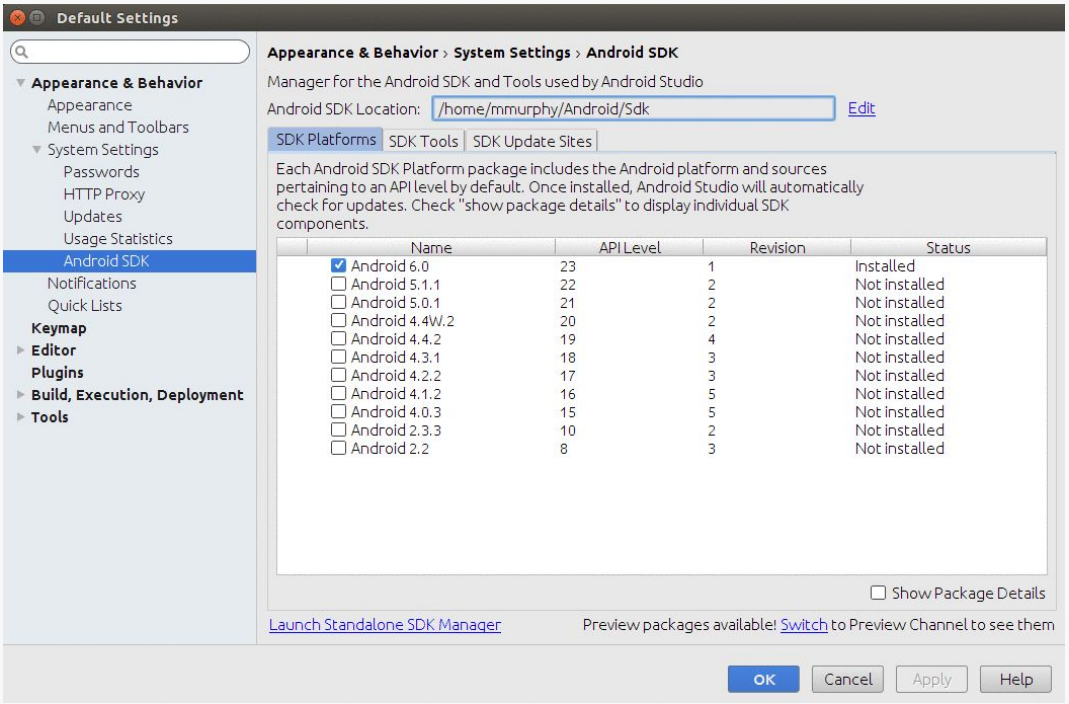


Figure 10: Android SDK Manager, "SDK Platforms" Tab

The "SDK Platforms" tab lists the versions of Android that you can compile against. The latest version of Android is usually installed when you set up Android Studio initially. However, for the tutorials, please also check "Android 4.4.2" in the list, and then click the "Apply" button to download and install that.

When that has completed, you will be returned to the SDK Manager. Click on the "SDK Tools" tab:

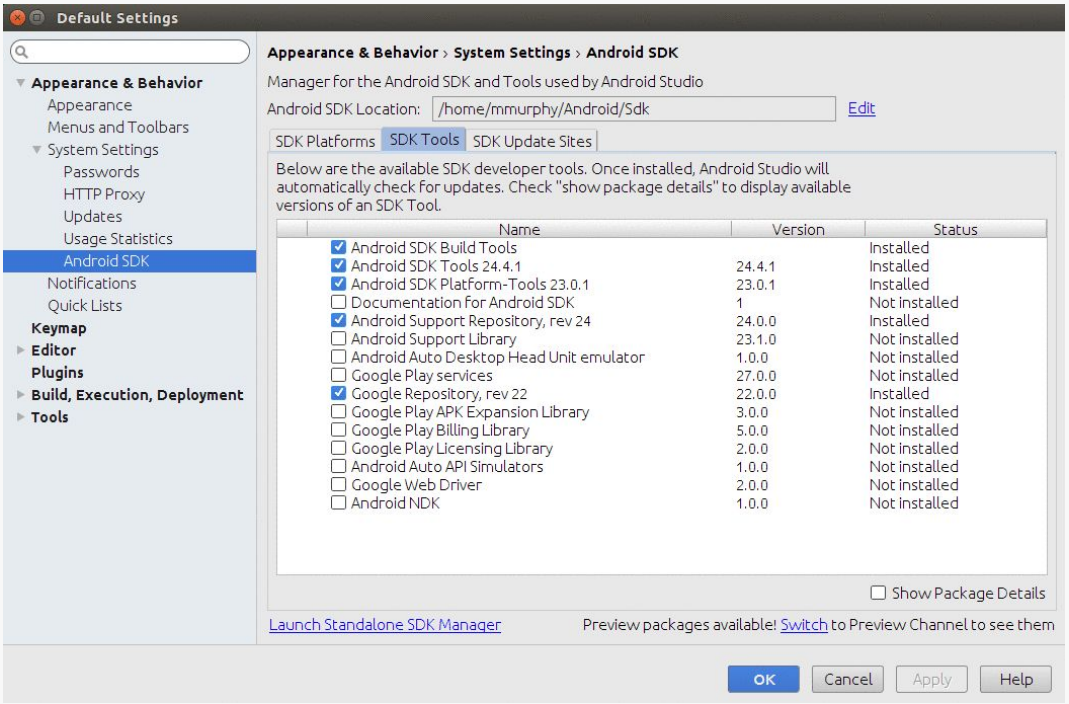


Figure 11: Android SDK Manager, "SDK Tools" Tab

This lists tools and related materials for Android development, other than the emulator (which is set up and configured separately). Android Studio usually has the right set up of stuff checked and installed already for you. You may wish to install the "Documentation for Android SDK", which amounts to an offline copy of most of the material found at `http://developer.android.com`. The other items in here are a bit more esoteric, and you will not be needing them for most of this book.

Some items may be marked with a status indicating that an update is available, in which case you may wish to apply those updates. Conversely, if anything labeled "preview" is checked, uncheck and uninstall it. Those items would be related to an outstanding developer preview of a new version of Android. While developer previews are useful, they add complexity for newcomers to Android.

When you are done making these adjustments, you can close up the SDK Manager by clicking the OK button.

In Our Next Episode…

… we will create an Android project that will serve as the basis for all our future tutorials, plus set up our emulator and device.