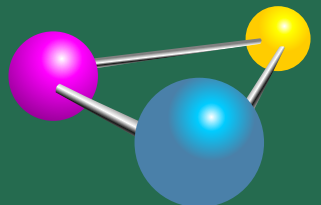


**Version
6.7**

*Supports Through
Android 5.1!*

The Busy Coder's Guide to Android Development

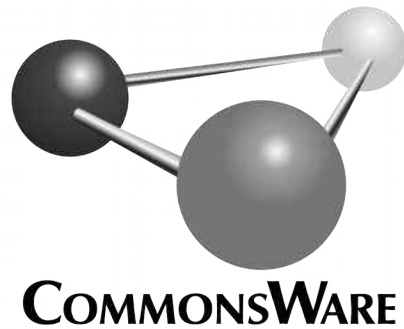
Mark L. Murphy



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

Copyright © 2008-2015 CommonsWare, LLC. All Rights Reserved.

Printed in the United States of America.

Printing History:

June 2015:

Version 6.7

ISBN: 978-0-9816780-0-9

The CommonsWare name and logo, “Busy Coder's Guide”, and related trade dress are trademarks of CommonsWare, LLC.

All other trademarks referenced in this book are trademarks of their respective firms.

The publisher and author(s) assume no responsibility for errors or omissions or for damages resulting from the use of the information contained herein.

Table of Contents

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

• Preface	
◦ Welcome to the Book!	37
◦ The Book's Structure	37
◦ <i>The Trails</i>	38
◦ About the Updates	43
◦ <i>Warescription</i>	43
◦ <i>Book Bug Bounty</i>	44
◦ Source Code and Its License	45
◦ <i>Creative Commons and the Four-to-Free (42F) Guarantee</i>	46
◦ Acknowledgments	46
• Key Android Concepts	
◦ Android Applications	1
◦ <i>Android Devices</i>	7
◦ Don't Be Scared	10
• Choosing Your Development Toolchain	
◦ Android Studio	11
◦ Eclipse	11
◦ <i>IntelliJ IDEA</i>	12
◦ Command-Line Builds via Gradle for Android	12
◦ Yet Other Alternatives	13
◦ 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
◦ <i>Step #3 - Install the Developer Tools</i>	16
◦ <i>Step #4 - Install the SDKs and Add-Ons</i>	20
◦ In Our Next Episode...	29
• Android and Projects	
◦ Common Concepts	31
◦ Projects and Android Studio	32
◦ Projects and Eclipse	39
◦ Starter Project Generators	45
• Tutorial #2 - Creating a Stub Project	
◦ About Our Tutorial Project	47

◦ About the Rest of the Tutorials	48
◦ About Our Tools	48
◦ Step #1: Creating the Project	49
◦ Step #2 - Set Up the Emulator	55
◦ Step #3 - Set Up the Device	67
◦ Step #4: Running the Project	70
◦ In Our Next Episode...	73
• Getting Around Android Studio	
◦ Navigating The Project Explorer	75
◦ Running Projects	78
◦ Viewing Output	79
◦ Accessing Android Tools	80
◦ Android Studio and Release Channels	84
◦ Visit the Trails!	85
• Contents of Android Projects	
◦ What You Get, In General	87
◦ The Contents of an Android Studio Project	89
◦ The Contents of an Eclipse Project	91
◦ What You Get Out Of It	92
• Introducing Gradle and the Manifest	
◦ Gradle: The Big Questions	93
◦ Obtaining Gradle	96
◦ Versions of Gradle and Gradle for Android	98
◦ Gradle Environment Variables	99
◦ Examining the Gradle Files	99
◦ Introducing the Manifest	101
◦ Things In Common Between the Manifest and Gradle	102
◦ Other Gradle Items of Note	105
◦ Where's the GUI?	107
◦ The Rest of the Manifest	107
◦ Learning More About Gradle	109
◦ Visit the Trails!	110
• Tutorial #3 - Changing Our Manifest (and Gradle File)	
◦ Some Notes About Relative Paths	111
◦ Step #1: Supporting Screens	112
◦ Step #2: Adding our Minimum and Target SDK Versions	115
◦ In Our Next Episode...	117
• Some Words About Resources	
◦ String Theory	120
◦ Got the Picture?	124
◦ Dimensions	129

◦ The Resource That Shall Not Be Named... Yet	131
• Tutorial #4 - Adjusting Our Resources	
◦ Step #1: Changing the Name	133
◦ Step #2: Changing the Icon	135
◦ Step #3: Running the Result	136
◦ In Our Next Episode...	138
• The Theory of Widgets	
◦ What Are Widgets?	139
◦ Size, Margins, and Padding	141
◦ What Are Containers?	141
◦ The Absolute Positioning Anti-Pattern	142
◦ The Theme of This Section: Themes	143
• The Android User Interface	
◦ The Activity	147
◦ Dissecting the Activity	148
◦ Using XML-Based Layouts	149
• Basic Widgets	
◦ Common Concepts	155
◦ Assigning Labels	158
◦ A Commanding Button	168
◦ Fleeting Images	173
◦ Fields of Green. Or Other Colors.	180
◦ More Common Concepts	186
◦ Visit the Trails!	188
• Debugging Your App	
◦ Get Thee To a Stack Trace	190
◦ The Case of the Confounding Class Cast	194
◦ Point Break	195
• LinearLayout and the Box Model	
◦ Concepts and Properties	197
◦ Android Studio Graphical Layout Editor	201
◦ Eclipse Graphical Layout Editor	202
• Other Common Widgets and Containers	
◦ Just a Box to Check	205
◦ Don't Like Checkboxes? How About Toggles or Switches?	210
◦ Turn the Radio Up	215
◦ All Things Are Relative	219
◦ Tabula Rasa	226
◦ Scrollwork	233
◦ Making Progress with ProgressBars	236
◦ Visit the Trails!	237

• Tutorial #5 - Making Progress	
◦ Step #1: Removing The “Hello, World”	239
◦ Step #2: Adding a ProgressBar	241
◦ Step #3: Seeing the Results	244
◦ In Our Next Episode...	244
• GUI Building, Continued	
◦ Making Your Selection	245
◦ Including Includes	246
◦ Wrap It Up (In a Container)	248
◦ Morphing Widgets	249
◦ Preview of Coming Attractions	250
• AdapterViews and Adapters	
◦ Adapting to the Circumstances	251
◦ Lists of Naughty and Nice	253
◦ Clicks versus Selections	255
◦ Spin Control	259
◦ Grid Your Lions (Or Something Like That...)	262
◦ Fields: Now With 35% Less Typing!	265
◦ Customizing the Adapter	269
◦ Visit the Trails!	277
• The WebView Widget	
◦ Role of WebView	279
◦ WebView and WebKit	280
◦ The Android System WebView	280
◦ Adding the Widget	281
◦ Loading Content Via a URL	281
◦ Links and Redirects	283
◦ Supporting JavaScript	283
◦ Alternatives for Loading Content	284
◦ Listening for Events	285
◦ Addressing the Link/Redirect Behavior	288
◦ Visit the Trails!	289
• Defining and Using Styles	
◦ Styles: DIY DRY	291
◦ Elements of Style	293
◦ Themes: Would a Style By Any Other Name...	296
◦ What Happens If You Have No Theme	296
• JARs and Library Projects	
◦ The Dalvik VM, and a Bit of ART	300
◦ Getting the Library	300
◦ The Outer Limits	302

◦ JAR Dependency Management	303
◦ OK, So What is a Library Project?	303
◦ Using a Library Project	304
◦ Library Projects: What You Get	305
◦ <i>The Android Support Package</i>	306
• Tutorial #6 - Adding a Library	
◦ Step #1: Attaching the Android Support Package	311
◦ Step #2: Downloading the Third-Party JARs	312
◦ In Our Next Episode...	314
• The Action Bar	
◦ Bar Hopping	315
◦ Yet Another History Lesson	321
◦ Your Action Bar Options	322
◦ Setting the Target	323
◦ Defining the Resource	325
◦ Applying the Resource	327
◦ Responding to Events	328
◦ The Rest of the Sample Activity	329
◦ MENU Key, We Hardly Knew Ye	335
◦ Action Bars, Live in Living Color!	336
◦ Visit the Trails!	345
• Tutorial #7 - Setting Up the Action Bar	
◦ <i>Step #1: Acquiring Some Icons</i>	347
◦ Step #2: Defining Some Options	348
◦ Step #3: Loading and Responding to Our Options	351
◦ In Our Next Episode...	355
• Android's Process Model	
◦ When Processes Are Created	357
◦ BACK, HOME, and Your Process	358
◦ Termination	359
◦ Foreground Means "I Love You"	360
◦ You and Your Heap	360
• Activities and Their Lifecycles	
◦ Creating Your Second (and Third and...) Activity	362
◦ Warning! Contains Explicit Intents!	368
◦ Using Implicit Intents	370
◦ Extra! Extra!	375
◦ Pondering Parcelable	377
◦ Asynchronicity and Results	378
◦ Schroedinger's Activity	378
◦ Life, Death, and Your Activity	379

◦ When Activities Die	381
◦ Walking Through the Lifecycle	382
◦ Recycling Activities	385
◦ Application: Transcending the Activity	386
• Tutorial #8 - Setting Up An Activity	
◦ Step #1: Creating the Stub Activity Class and Manifest Entry	389
◦ Step #2: Launching Our Activity	392
◦ In Our Next Episode...	393
• The Tactics of Fragments	
◦ The Six Questions	395
◦ Where You Get Your Fragments From	398
◦ Your First Fragment	398
◦ The Fragment Lifecycle Methods	401
◦ Your First Dynamic Fragment	402
◦ Fragments and the Action Bar	405
◦ Fragments Within Fragments: Just Say “Maybe”	406
◦ Fragments and Multiple Activities	407
• Tutorial #9 - Starting Our Fragments	
◦ Step #1: Create a SimpleContentFragment	409
◦ Step #2: Examining SimpleContentFragment	412
◦ In Our Next Episode...	412
• Swiping with ViewPager	
◦ Swiping Design Patterns	413
◦ Pieces of a Pager	414
◦ Paging Fragments	414
◦ Paging Other Stuff	419
◦ Indicators	420
• Tutorial #10 - Rigging Up a ViewPager	
◦ Step #1: Add a ViewPager to the Layout	425
◦ Step #2: Obtaining Our ViewPager	426
◦ Step #3: Creating a ContentsAdapter	427
◦ Step #4: Setting Up the ViewPager	428
◦ In Our Next Episode...	429
• Resource Sets and Configurations	
◦ What’s a Configuration? And How Do They Change?	431
◦ Configurations and Resource Sets	432
◦ Screen Size and Orientation	433
◦ Coping with Complexity	436
◦ Choosing The Right Resource	437
◦ API-Versioned Resources	441
◦ Default Change Behavior	443

◦ State Saving Scenarios	444
◦ <i>Your Options for Configuration Changes</i>	445
◦ Blocking Rotations	457
◦ <i>And Now, a Word From the Android Project View</i>	457
• Material Design Basics	
◦ Your App, in Technicolor!	461
• Dealing with Threads	
◦ The Main Application Thread	469
◦ Getting to the Background	471
◦ <i>Asyncing Feeling</i>	471
◦ Alternatives to AsyncTask	481
◦ And Now, The Caveats	482
◦ <i>Event Buses</i>	483
◦ Visit the Trails!	490
• Requesting Permissions	
◦ Mother, May I?	492
◦ New Permissions in Old Applications	494
◦ Permissions: Up Front Or Not At All	494
◦ Signature Permissions	495
◦ Requiring Permissions	496
• Assets, Files, and Data Parsing	
◦ Packaging Files with Your App	497
◦ Files and Android	499
◦ Working with Internal Storage	500
◦ Working with External Storage	503
◦ Multiple User Accounts	507
◦ Linux Filesystems: You Sync, You Win	508
◦ StrictMode: Avoiding Janky Code	509
◦ XML Parsing Options	512
◦ <i>JSON Parsing Options</i>	513
◦ Visit the Trails!	513
• Tutorial #11 - Adding Simple Content	
◦ Step #1: Adding Some Content	515
◦ Step #2: Using SimpleContentFragment	516
◦ Step #3: Launching Our Activities, For Real This Time	517
◦ In Our Next Episode...	519
• Tutorial #12 - Displaying the Book	
◦ Step #1: Adding a Book	521
◦ Step #2: Creating a ModelFragment	522
◦ Step #3: Defining Our Model	523
◦ Step #4: Examining Our Model	525

◦ Step #5: Defining Our Event	525
◦ Step #6: Loading Our Model	527
◦ Step #7: Registering for Events	529
◦ Step #8: Adapting the Content	530
◦ Step #9: Showing the Content When Loaded	530
◦ Step #10: Attaching our ModelFragment	531
◦ Step #11: Showing the Content After a Configuration Change	532
◦ Step #12: Going Home, Again	533
◦ Step #13: Setting Up StrictMode	534
◦ In Our Next Episode...	535
• Using Preferences	
◦ Getting What You Want	537
◦ Stating Your Preference	538
◦ Introducing PreferenceActivity	539
◦ Types of Preferences	549
◦ Intents for Headers or Preferences	552
◦ Conditional Headers	553
• Tutorial #13 - Using Some Preferences	
◦ Step #1: Defining the Preference XML Files	559
◦ Step #2: Creating Our PreferenceActivity	562
◦ Step #3: Adding To Our Action Bar	564
◦ Step #4: Launching the PreferenceActivity	564
◦ Step #5: Loading the Preferences	568
◦ Step #6: Saving the Last-Read Position	571
◦ Step #7: Restoring the Last-Read Position	572
◦ Step #8: Keeping the Screen On	573
◦ In Our Next Episode...	575
• SQLite Databases	
◦ Introducing SQLite	577
◦ Thinking About Schemas	578
◦ Start with a Helper	578
◦ Getting Data Out	583
◦ The Rest of the CRUD	589
◦ Hey, What About Hibernate?	595
◦ Visit the Trails!	595
• Tutorial #14 - Saving Notes	
◦ Step #1: Adding a DatabaseHelper	597
◦ Step #2: Examining DatabaseHelper	599
◦ Step #3: Creating a NoteFragment	599
◦ Step #4: Examining NoteFragment	604
◦ Step #5: Creating the NoteActivity	604

◦ Step #6: Examining NoteActivity	606
◦ Step #7: Add Notes to the Action Bar	606
◦ Step #8: Defining a NoteLoadedEvent	608
◦ Step #9: Loading a Note from the Database	609
◦ Step #10: Loading the Note Into the Fragment	610
◦ Step #11: Updating the Database	611
◦ Step #12: Saving the Note	612
◦ Step #13: Adding a Delete Action Bar Item	614
◦ Step #14: Closing the NoteFragment When Deleted	615
◦ In Our Next Episode...	620
• Internet Access	
◦ DIY HTTP	621
◦ HTTP via DownloadManager	631
◦ Using Third-Party JARs	631
◦ SSL	632
◦ Using HTTP Client Libraries	632
◦ Visit the Trails	643
• Intents, Intent Filters	
◦ What's Your Intent?	645
◦ Stating Your Intent(ions)	647
◦ Responding to Implicit Intents	647
◦ Requesting Implicit Intents	650
◦ ShareActionProvider	654
• Broadcasts and Broadcast Receivers	
◦ The Stopped State	659
◦ Example System Broadcasts	660
◦ The Order of Things	668
◦ Keeping It Local	669
◦ Visit the Trails!	670
• Tutorial #15 - Sharing Your Notes	
◦ Step #1: Adding a ShareActionProvider	671
◦ Step #2: Sharing the Note	672
◦ Step #3: Testing the Result	674
◦ In Our Next Episode...	675
• Services and the Command Pattern	
◦ Why Services?	677
◦ Setting Up a Service	678
◦ Communicating To Services	680
◦ Scenario: The Music Player	682
◦ Communicating From Services	685
◦ Scenario: The Downloader	687

◦ <i>Services and Configuration Changes</i>	692
• Tutorial #16 - Updating the Book	
◦ Step #1: Adding a Stub DownloadCheckService	693
◦ Step #2: Tying the Service Into the Action Bar	695
◦ Step #3: Defining Our Event	696
◦ Step #4: Defining Our JSON	696
◦ Step #5: Defining Our Retrofit Interface	698
◦ Step #6: Retrieving Our JSON Via Retrofit	699
◦ Step #7: Downloading the Update	700
◦ Step #8: Unpacking the Update	702
◦ Step #9: Using the Update	706
◦ In Our Next Episode...	710
• AlarmManager and the Scheduled Service Pattern	
◦ Scenarios	711
◦ Options	712
◦ A Simple Example	714
◦ The Five set...() Varieties	716
◦ The Four Types of Alarms	718
◦ When to Schedule Alarms	718
◦ Archetype: Scheduled Service Polling	720
◦ Staying Awake at Work	723
◦ Warning: Not All Android Devices Play Nice	728
◦ Debugging Alarms	728
◦ WakefulBroadcastReceiver	731
• Tutorial #17 - Periodic Book Updates	
◦ Step #1: Adding a Stub UpdateReceiver	735
◦ Step #2: Scheduling the Alarms	737
◦ Step #3: Adding the WakefulIntentService	739
◦ Step #4: Using WakefulIntentService	740
◦ Step #5: Completing the UpdateReceiver	740
◦ In Our Next Episode...	741
• Notifications	
◦ What's a Notification?	743
◦ Showing a Simple Notification	746
◦ The Activity-Or-Notification Scenario	750
◦ Big (and Rich) Notifications	751
◦ Foreground Services	758
◦ Disabled Notifications	761
• Tutorial #18 - Notifying the User	
◦ Step #1: Raising the Notification	765
◦ In Our Next Episode...	767

• Large-Screen Strategies and Tactics	
◦ Objective: Maximum Gain, Minimum Pain	769
◦ <i>The Fragment Strategy</i>	769
◦ <i>Fragment Example: The List-and-Detail Pattern</i>	778
◦ Other Master-Detail Strategies	790
◦ Showing More Pages	802
◦ Fragment FAQs	807
◦ Screen Size and Density Tactics	808
◦ Other Considerations	812
• Tutorial #19 - Supporting Large Screens	
◦ Step #1: Creating Our Layouts	815
◦ Step #2: Loading Our Sidebar Widgets	821
◦ Step #3: Opening the Sidebar	822
◦ Step #4: Loading Content Into the Sidebar	822
◦ Step #5: Removing Content From the Sidebar	825
• Backwards Compatibility Strategies and Tactics	
◦ Think Forwards, Not Backwards	831
◦ Aim Where You Are Going	833
◦ A Target-Rich Environment	833
◦ Lint: It's Not Just For Belly Buttons	834
◦ A Little Help From Your Friends	835
◦ Avoid the New on the Old	835
◦ Testing	839
◦ Keeping Track of Changes	839
• Getting Help	
◦ Questions. Sometimes, With Answers.	841
◦ Heading to the Source	842
◦ Getting Your News Fix	843
• Working with Library Projects	
◦ Prerequisites	845
◦ Creating a Library Project	845
◦ Using a Library Project, Part II	849
◦ Library Projects and the Manifest	849
◦ Limitations of Library Projects	850
• Gradle and Eclipse Projects	
◦ Prerequisites and Warnings	851
◦ "Legacy"?	851
◦ Creating Your Gradle Build File	852
◦ Examining the Gradle File	857
• Gradle and Tasks	
◦ Key Build-Related Tasks	859

◦ Results	861
• Gradle and the New Project Structure	
◦ Prerequisites and Warnings	863
◦ Objectives of the New Project Structure	864
◦ Terminology	864
◦ Creating a Project in the New Structure	868
◦ What the New Project Structure Looks Like	868
◦ Configuring the Stock Build Types	871
◦ Adding Build Types	876
◦ Adding Product Flavors and Getting Build Variants	877
◦ Doing the Splits	882
◦ Revisiting the Legacy Gradle File	884
◦ Working with the New Project Structure in Android Studio	885
◦ Flavors, Build Types, and the Project Structure Dialog	888
• Gradle and Dependencies	
◦ Prerequisites and Warnings	889
◦ “Dependencies”?	890
◦ A Tale of Two Dependencies Closures	890
◦ Depending Upon a JAR	890
◦ Depending Upon NDK Binaries	892
◦ Depending Upon an Android Library Project	892
◦ Depending Upon Sub-Projects	894
◦ <i>Depending Upon Artifacts</i>	895
◦ Creating Android JARs from Gradle	902
◦ A Property of Transitive (Dependencies)	903
◦ Dependencies By Build Type	904
◦ Dependencies By Flavor	904
◦ Examining Some CWAC Builds	905
◦ Dependencies and the Project Structure Dialog	911
• Manifest Merger Rules	
◦ Prerequisites	913
◦ Manifest Scenarios	914
◦ Pieces of Manifest Generation	915
◦ Examining the Merger Results	917
◦ Merging Elements and Attributes	918
◦ Employing Placeholders	925
• Signing Your App	
◦ Prerequisites	929
◦ Role of Code Signing	929
◦ What Happens In Debug Mode	930
◦ Production Signing Keys	931

• Distribution	
◦ Prerequisites	947
◦ <i>Get Ready To Go To Market</i>	947
• Advanced Gradle for Android Tips	
◦ Prerequisites	953
◦ Gradle, DRY	953
◦ Automating APK Version Information	959
◦ Adding to BuildConfig	961
◦ Down and Dirty with the DSL	963
• JUnit and Android	
◦ Prerequisites	965
◦ JUnit Basics	965
◦ Pondering Gradle for Android	966
◦ Where Your Test Code Lives	967
◦ Where Your Test Code Runs	971
◦ <i>Writing Your Test Cases</i>	972
◦ Your Test Suite	977
◦ Running Your Tests	977
◦ Testing Android Library Projects	984
◦ Test Dependencies	985
◦ Testing Legacy Project Structures with Gradle for Android	986
• Testing with JUnit4	
◦ Prerequisites	989
◦ <i>The AndroidJUnitRunner</i>	989
◦ <i>JUnit4</i>	992
• MonkeyRunner and the Test Monkey	
◦ Prerequisites	1003
◦ <i>MonkeyRunner</i>	1003
◦ <i>Monkeying Around</i>	1005
• Testing with UI Automator	
◦ Prerequisites	1009
◦ What Is UI Automator?	1009
◦ <i>Why Choose UI Automator Over Alternatives?</i>	1010
◦ <i>Gradle and Android Studio Settings</i>	1010
◦ Creating a Test Case	1011
◦ Running Your Tests	1018
◦ Finding Your Widgets	1019
• Introducing GridLayout	
◦ Prerequisites	1021
◦ Issues with the Classic Containers	1021
◦ The New Contender: GridLayout	1023

◦ GridLayout and the Android Support Package	1023
◦ Eclipse and GridLayout	1025
◦ Trying to Have Some Rhythm	1025
◦ Our Test App	1026
◦ Replacing the Classics	1028
◦ Implicit Rows and Columns	1034
◦ Row and Column Spans	1036
• Dialogs and DialogFragments	
◦ Prerequisites	1041
◦ DatePickerDialog and TimePickerDialog	1041
◦ AlertDialog	1047
◦ DialogFragments	1048
◦ DialogFragment: The Other Flavor	1052
◦ Dialogs: Modal, Not Blocking	1053
• Advanced ListViews	
◦ Prerequisites	1055
◦ Multiple Row Types, and Self Inflation	1055
◦ Choice Modes and the Activated Style	1061
◦ Custom Mutable Row Contents	1062
◦ From Head To Toe	1067
◦ Enter RecyclerView	1071
• Action Bar Navigation	
◦ Prerequisites	1073
◦ List Navigation	1073
◦ Tabs (And Sometimes List) Navigation	1078
◦ Custom Navigation	1084
• Action Modes	
◦ Prerequisites	1086
◦ A Matter of Context	1086
◦ Manual Action Modes	1087
◦ Multiple-Choice-Modal Action Modes	1092
◦ Long-Click To Initiate an Action Mode	1096
• Other Advanced Action Bar Techniques	
◦ Prerequisites	1103
◦ Splitting the Bar	1103
◦ Action Views and Action Providers	1107
◦ Searching with SearchView	1108
◦ Floating Action Bars	1113
• AppCompat: The Official Action Bar Backport	
◦ Prerequisites	1117
◦ <i>Ummmm... Why?</i>	1117

◦	<i>The Basics of Using AppCompatActivity</i>	1120
◦	<i>Other AppCompatActivity Effects</i>	1127
◦	<i>And Then, There Are the Bugs</i>	1133
◦	<i>To Material, or Not to Material</i>	1133
•	ActionBarSherlock	
◦	Prerequisites	1137
◦	Installation	1138
◦	Code Changes	1138
•	RecyclerView	
◦	Prerequisites	1142
◦	AdapterView and its Discontents	1142
◦	Enter RecyclerView	1142
◦	<i>A Trivial List</i>	1143
◦	<i>Divider Options</i>	1151
◦	<i>Handling Click Events</i>	1157
◦	What About Cursors?	1163
◦	Grids	1167
◦	<i>Varying the Items</i>	1170
◦	<i>Mutable Row Contents</i>	1183
◦	<i>Changing the Contents</i>	1202
◦	<i>The Order of Things</i>	1208
◦	Other Bits of Goodness	1216
◦	<i>The March of the Libraries</i>	1217
•	Implementing a Navigation Drawer	
◦	Prerequisites	1221
◦	What is a Navigation Drawer?	1221
◦	A Simple Navigation Drawer	1223
◦	<i>Alternative Row Layouts</i>	1229
◦	Additional Considerations	1231
◦	What Should Not Be in the Drawer	1240
◦	Independent Implementations	1240
•	Advanced Uses of WebView	
◦	Prerequisites	1243
◦	Friends with Benefits	1243
◦	Turnabout is Fair Play	1249
◦	Navigating the Waters	1253
◦	Settings, Preferences, and Options (Oh, My!)	1253
◦	<i>Security and Your WebView</i>	1254
•	The Input Method Framework	
◦	Prerequisites	1259
◦	Keyboards, Hard and Soft	1259

◦ Tailored To Your Needs	1260
◦ Tell Android Where It Can Go	1265
◦ Fitting In	1267
◦ Jane, Stop This Crazy Thing!	1269
• Fonts	
◦ Prerequisites	1271
◦ Love The One You're With	1271
◦ Yeah, But Do We Really Have To Do This in Java?	1275
◦ Here a Glyph, There a Glyph	1276
• Rich Text	
◦ Prerequisites	1279
◦ The Span Concept	1279
◦ Loading Rich Text	1281
◦ Editing Rich Text	1284
◦ Saving Rich Text	1288
◦ Manipulating Rich Text	1289
• Animators	
◦ Prerequisites	1291
◦ ViewPropertyAnimator	1291
◦ The Foundation: Value and Object Animators	1296
◦ Animating Custom Types	1298
◦ Hardware Acceleration	1300
◦ The Three-Fragment Problem	1301
• Legacy Animations	
◦ Prerequisites	1313
◦ It's Not Just For Toons Anymore	1313
◦ A Quirky Translation	1314
◦ Fading To Black. Or Some Other Color.	1318
◦ When It's All Said And Done	1320
◦ Loose Fill	1321
◦ Hit The Accelerator	1321
◦ Animate. Set. Match.	1322
◦ Active Animations	1323
• Custom Drawables	
◦ Prerequisites	1325
◦ Where Do These Things Go?	1326
◦ ColorDrawable	1326
◦ AnimationDrawable	1326
◦ StateListDrawable	1330
◦ ColorStateList	1331
◦ LayerDrawable	1333

◦ TransitionDrawable	1334
◦ LevelListDrawable	1335
◦ ScaleDrawable and ClipDrawable	1336
◦ InsetDrawable	1345
◦ ShapeDrawable	1346
◦ VectorDrawable	1356
◦ BitmapDrawable	1356
◦ Composite Drawables	1364
◦ <i>A Stitch In Time Saves Nine</i>	1367
• Mapping with Maps V2	
◦ Prerequisites	1377
◦ A Brief History of Mapping on Android	1378
◦ Where You Can Use Maps V2	1378
◦ Licensing Terms for Maps V2	1379
◦ What You Need to Start	1379
◦ The Book Samples... And You!	1383
◦ <i>Setting Up a Basic Map</i>	1384
◦ Playing with the Map	1389
◦ Map Tiles	1392
◦ Placing Simple Markers	1392
◦ Seeing All the Markers	1395
◦ Flattening and Rotating Markers	1397
◦ Sprucing Up Your “Info Windows”	1401
◦ Images and Your Info Window	1406
◦ Setting the Marker Icon	1411
◦ Responding to Taps	1413
◦ Dragging Markers	1415
◦ The “Final” Limitations	1417
◦ A Bit More About IPC	1419
◦ Finding the User	1420
◦ Drawing Lines and Areas	1425
◦ Gestures and Controls	1428
◦ Tracking Camera Changes	1428
◦ Maps in Fragments and Pagers	1430
◦ Animating Marker Movement	1435
◦ Maps, of the Indoor Variety	1444
◦ Taking a Snapshot of a Map	1444
◦ MapFragment vs. MapView	1445
◦ About That AbstractMapActivity Class... ..	1446
◦ Helper Libraries for Maps V2	1450
◦ Problems with Maps V2 at Runtime	1454

◦ Problems with Maps V2 Deployment	1454
◦ What Non-Compliant Devices Show	1454
◦ Mapping Alternatives	1455
◦ News and Getting Help	1455
• Crafting Your Own Views	
◦ Prerequisites	1457
◦ Pick Your Poison	1457
◦ Colors, Mixed How You Like Them	1458
◦ ReverseChronometer: Simply a Custom Subclass	1468
◦ AspectLockedFrameLayout: A Custom Container	1474
◦ Mirror and MirroringFrameLayout: Draw It Yourself	1478
• Custom Dialogs and Preferences	
◦ Prerequisites	1489
◦ Your Dialog, Chocolate-Covered	1489
◦ Preferring Your Own Preferences, Preferably	1493
• Progress Indicators	
◦ Prerequisites	1501
◦ Progress Bars	1501
◦ ProgressBar and Threads	1504
◦ Tailoring Progress Bars	1507
◦ Progress Dialogs	1515
◦ Title Bar and Action Bar Progress Indicators	1517
◦ Direct Progress Indication	1519
• Advanced Notifications	
◦ Prerequisites	1521
◦ Being a Good Citizen	1521
◦ Wear? There!	1522
◦ Stacking Notifications	1527
◦ Avoiding Wear	1533
◦ Other Wear-Specific Notification Options	1534
◦ Lockscreen Notifications	1547
◦ Priority, and Heads-Up Notifications	1556
◦ Full-Screen Notifications	1558
◦ Custom Views	1561
◦ Seeing It In Action	1562
◦ Progress Notifications	1568
◦ Life After Delete	1571
◦ The Mysterious Case of the Missing Number	1572
• More Fun with Pagers	
◦ Prerequisites	1573
◦ Hosting ViewPager in a Fragment	1573

◦ Pages and the Action Bar	1575
◦ ViewPagers and Scrollable Contents	1577
◦ Using ViewPagerIndicator	1578
◦ Columns for Large, Pages for Small	1582
◦ Introducing ArrayAdapter	1588
◦ Columns for Large Landscape, Pages for the Rest	1591
◦ Adding, Removing, and Moving Pages	1596
◦ Inside ArrayAdapter	1600
• Focus Management and Accessibility	
◦ Prerequisites	1613
◦ Prepping for Testing	1614
◦ Controlling the Focus	1614
◦ Accessibility and Focus	1623
◦ Accessibility Beyond Focus	1623
◦ Accessibility Beyond Impairment	1633
• Miscellaneous UI Tricks	
◦ Prerequisites	1635
◦ Full-Screen and Lights-Out Modes	1635
◦ Offering a Delayed Timeout	1646
• Event Bus Alternatives	
◦ Prerequisites	1651
◦ A Brief Note About the Sample Apps	1651
◦ Standard Intents as Event Bus	1651
◦ LocalBroadcastManager as Event Bus	1652
◦ Square's Otto	1662
◦ Revisiting greenrobot's EventBus	1668
• Tasks	
◦ Prerequisites	1673
◦ First, Some Terminology	1673
◦ And Now, a Bit About Task Killers	1679
◦ A Canary for the Task's Coal Mine	1686
◦ The Default User Experience	1688
◦ Explaining the Default Behavior	1692
◦ Basic Scenarios for Changing the Behavior	1696
◦ Dealing with the Persistent Tasks	1710
◦ Documents As Tasks	1712
◦ Other Task-Related Activity Properties	1715
◦ Other Task-Related Activity Methods	1723
• Home Screen App Widgets	
◦ Prerequisites	1727
◦ App Widgets and Security	1727

◦ <i>The Big Picture for a Small App Widget</i>	1728
◦ <i>Crafting App Widgets</i>	1729
◦ Another and Another	1736
◦ <i>App Widgets: Their Life and Times</i>	1736
◦ Controlling Your (App Widget's) Destiny	1737
◦ One Size May Not Fit All	1737
◦ <i>Lockscreen Widgets</i>	1744
◦ <i>Preview Images</i>	1750
◦ <i>Being a Good Host</i>	1752
• Adapter-Based App Widgets	
◦ Prerequisites	1753
◦ AdapterViews for App Widgets	1753
◦ <i>Building Adapter-Based App Widgets</i>	1754
• Content Provider Theory	
◦ Prerequisites	1769
◦ <i>Using a Content Provider</i>	1769
◦ <i>Building Content Providers</i>	1777
◦ <i>Issues with Content Providers</i>	1785
• Content Provider Implementation Patterns	
◦ Prerequisites	1787
◦ The Single-Table Database-Backed Content Provider	1787
◦ The Local-File Content Provider	1794
◦ The Protected Provider	1802
◦ The Stream Provider	1805
◦ FileProvider	1808
◦ StreamProvider	1812
• The Loader Framework	
◦ <i>Prerequisites</i>	1815
◦ Cursors: Issues with Management	1816
◦ Introducing the Loader Framework	1816
◦ Honeycomb... Or Not	1818
◦ Using CursorLoader	1818
◦ What Else Is Missing?	1820
◦ Issues, Issues, Issues	1821
◦ Loaders Beyond Cursors	1821
◦ What Happens When...?	1821
• The ContactsContract and CallLog Providers	
◦ <i>Prerequisites</i>	1825
◦ Introducing You to Your Contacts	1826
◦ <i>Pick a Peck of Pickled People</i>	1827
◦ Spin Through Your Contacts	1829

◦ Makin' Contacts	1835
◦ <i>Looking at the CallLog</i>	1838
• The CalendarContract Provider	
◦ <i>Prerequisites</i>	1844
◦ You Can't Be a Faker	1844
◦ Do You Have Room on Your Calendar?	1844
◦ Penciling In an Event	1849
• The MediaStore Provider	
◦ <i>Prerequisites</i>	1851
◦ What Is the MediaStore?	1852
◦ MediaStore and "Other" External Storage	1853
◦ How Does My Content Get Indexed?	1854
◦ How Do I Retrieve Video from the MediaStore?	1854
• Consuming Documents	
◦ <i>Prerequisites</i>	1863
◦ The Storage Access... What?	1863
◦ The Storage Access Framework Participants	1865
◦ Picking How to Pick (a Peck of Pickled Pepper Photos)	1865
◦ Opening a Document	1866
◦ The Rest of the CRUD	1869
◦ Pondering Persistent Permissions	1870
• Providing Documents	
◦ Prerequisites	1873
◦ Have Your Content, and Provide it Too	1873
◦ Key Provider Concepts	1875
◦ Pieces of a Provider	1876
◦ Optional Provider Capabilities	1890
• Encrypted Storage	
◦ <i>Prerequisites</i>	1898
◦ Scenarios for Encryption	1898
◦ Obtaining SQLCipher	1899
◦ Attaching SQLCipher To Your Project	1899
◦ Using SQLCipher	1900
◦ SQLCipher Limitations	1903
◦ Passwords and Sessions	1904
◦ <i>About Those Passphrases...</i>	1904
◦ Encrypted Preferences	1911
◦ IOCipher	1914
• Packaging and Distributing Data	
◦ <i>Prerequisites</i>	1915
◦ Packing a Database To Go	1915

• Advanced Database Techniques	
◦ Prerequisites	1919
◦ Full-Text Indexing	1919
• Miscellaneous Network Capabilities	
◦ Prerequisites	1935
◦ Downloading Files	1935
• Audio Playback	
◦ Prerequisites	1949
◦ Get Your Media On	1949
◦ MediaPlayer for Audio	1950
◦ Other Ways to Make Noise	1956
• Audio Recording	
◦ Prerequisites	1959
◦ Recording by Intent	1959
◦ Recording to Files	1962
◦ Recording to Streams	1965
◦ Raw Audio Input	1968
◦ Requesting the Microphone	1968
• Video Playback	
◦ Prerequisites	1969
◦ Moving Pictures	1969
• Using the Camera via 3rd-Party Apps	
◦ Prerequisites	1975
◦ Being Specific About Features	1975
◦ Still Photos: Letting the Camera App Do It	1976
◦ Scanning with ZXing	1978
◦ Videos: Letting the Camera App Do It	1979
◦ Directly Working with the Camera	1981
• Working Directly with the Camera	
◦ Prerequisites	1984
◦ Basic CameraFragment Usage	1984
◦ Simple Configuration and Usage	1985
◦ Core Camera Concepts	1992
◦ Advanced CWAC-Camera Features	2014
• Media Routes	
◦ Prerequisites	2023
◦ Terminology	2023
◦ A Tale of Three MediaRouters	2024
◦ Attaching to MediaRouter	2025
◦ User Route Selection with MediaRouteActionProvider	2027
◦ Using Live Audio Routes	2044

◦ Using Live Video Routes	2044
◦ Using Remote Playback Routes	2044
• Supporting External Displays	
◦ Prerequisites	2063
◦ A History of External Displays	2063
◦ What is a Presentation?	2064
◦ Playing with External Displays	2065
◦ Detecting Displays	2071
◦ <i>A Simple Presentation</i>	2072
◦ A Simpler Presentation	2077
◦ Presentations and Configuration Changes	2082
◦ <i>Presentations as Fragments</i>	2083
◦ Another Sample Project: Slides	2093
◦ Device Support for Presentation	2100
◦ Presentations from a Service	2101
◦ Hey, What About Chromecast?	2104
• Google Cast and Chromecast	
◦ Prerequisites	2105
◦ Here a Cast, There a Cast	2105
◦ Common Chromecast Development Notes	2107
◦ Your API Choices	2107
◦ Senders and Receivers	2108
◦ Supported Media Types	2109
◦ Cast SDK Dependencies	2110
◦ Developing Google Cast Apps	2112
• The “Ten-Foot UI”	
◦ Prerequisites	2113
◦ What is the “Ten-Foot UI”?	2114
◦ Overscan	2114
◦ Navigation	2115
◦ Stylistic Considerations	2116
◦ The Leanback UI	2118
◦ Testing Your Theories	2132
• Putting the TVs All Together: Decktastic	
◦ Prerequisites	2136
◦ Introducing Decktastic	2136
◦ Trying Decktastic Yourself	2140
◦ Implementing Decktastic	2140
• Creating a MediaRouteProvider	
◦ Prerequisites	2171
◦ Terminology	2171

◦	DIY Chromecast	2172
◦	Creating the MediaRouteProvider	2174
◦	Consuming the MediaRouteProvider	2184
◦	Implementing This “For Realz”	2187
•	SSL	
◦	Prerequisites	2191
◦	Basic SSL Operation	2191
◦	Common SSL Problems	2192
◦	TrustManagers	2193
◦	TrustManagerBuilder	2195
◦	About That Man in the Middle	2199
◦	Self-Signed Certificates, Revisited	2201
◦	Certificate Memorizing	2201
◦	Pinning	2206
◦	NetCipher	2206
•	Advanced Permissions	
◦	Prerequisites	2209
◦	Securing Yourself	2209
◦	Signature Permissions	2212
◦	The Custom Permission Vulnerability	2214
•	Restricted Profiles and UserManager	
◦	Prerequisites	2225
◦	Android Tablets and Multiple User Accounts	2225
◦	Determining What the User Can Do	2231
◦	Impacts of Device-Level Restrictions	2233
◦	Enabling Custom Restrictions	2234
◦	Implicit Intents May Go “Boom”	2244
•	Tapjacking	
◦	Prerequisites	2247
◦	What is Tapjacking?	2247
◦	Detecting Potential Tapjackers	2252
◦	Defending Against Tapjackers	2254
◦	Why Is This Being Discussed?	2257
◦	What Changed in 4.0.3?	2258
•	Miscellaneous Security Techniques	
◦	Prerequisites	2259
◦	Public Key Validation	2259
◦	Choosing Your Signing Keysize	2275
◦	Avoiding Accidental APIs	2276
◦	Other Ways to Expose Data	2281
•	Accessing Location-Based Services	

◦ Prerequisites	2285
◦ Location Providers: They Know Where You're Hiding	2286
◦ Finding Yourself	2286
◦ On the Move	2288
◦ Are We There Yet? Are We There Yet? Are We There Yet?	2289
◦ Testing... Testing...	2290
◦ <i>Alternative Flavors of Updates</i>	2291
◦ The Fused Option	2292
• The Fused Location Provider	
◦ Prerequisites	2293
◦ Why Use the Fused Location Provider?	2293
◦ Why Not Use the Fused Location Provider?	2294
◦ Finding Our Location, Once	2294
◦ Requesting Location Updates	2302
◦ I Can Haz Location?	2304
• Working with the Clipboard	
◦ Prerequisites	2313
◦ Using the Clipboard on Android 1.x/2.x	2313
◦ Advanced Clipboard on Android 3.x and Higher	2316
◦ Monitoring the Clipboard	2321
◦ The Android 4.3 Clipboard Bug	2323
• Telephony	
◦ Prerequisites	2325
◦ Report To The Manager	2326
◦ You Make the Call!	2326
◦ No, Really, You Make the Call!	2329
• Working With SMS	
◦ Prerequisites	2331
◦ Sending Out an SOS, Give or Take a Letter	2332
◦ Monitoring and Receiving SMS	2338
◦ The SMS Inbox	2345
◦ Asking to Change the Default	2346
◦ SMS and the Emulator	2347
• NFC	
◦ Prerequisites	2349
◦ What Is NFC?	2349
◦ To NDEF, Or Not to NDEF	2351
◦ NDEF Modalities	2351
◦ NDEF Structure and Android's Translation	2352
◦ The Reality of NDEF	2353
◦ Sources of Tags	2355

◦ Writing to a Tag	2355
◦ Responding to a Tag	2363
◦ Expected Pattern: Bootstrap	2364
◦ Mobile Devices are Mobile	2365
◦ Enabled and Disabled	2365
◦ Android Beam	2366
◦ Beaming Files	2372
◦ Another Sample: SecretAgentMan	2374
◦ Additional Resources	2382
• Device Administration	
◦ Prerequisites	2385
◦ Objectives and Scope	2385
◦ Defining and Registering an Admin Component	2386
◦ Going Into Lockdown	2392
◦ Passwords and Device Administration	2399
◦ Getting Along with Others	2403
• PowerManager and WakeLocks	
◦ Prerequisites	2405
◦ Keeping the Screen On, UI-Style	2405
◦ The Role of the WakeLock	2406
◦ What WakefulIntentService Does	2407
• JobScheduler	
◦ Prerequisites	2409
◦ The Limitations of AlarmManager	2409
◦ Enter the JobScheduler	2410
◦ Employing JobScheduler	2410
◦ Pondering Backoff Criteria	2423
◦ Other JobScheduler Features	2424
• Push Notifications with GCM	
◦ Prerequisites	2427
◦ The Precursor: C2DM	2428
◦ The Replacement: GCM	2428
◦ The Re-Replacement: GCM 2013	2428
◦ The Pieces of Push	2429
◦ A Simple Push	2435
◦ Message Options and Advanced Features	2449
◦ Re-Registration	2451
◦ Additional Features	2452
◦ Considering Encryption	2454
◦ Issues with GCM	2454
◦ Amazon Simple Notification Service and GCM	2456

• Basic Use of Sensors	
◦ Prerequisites	2457
◦ The Sensor Abstraction Model	2457
◦ Considering Rates	2458
◦ Reading Sensors	2459
◦ Batching Sensor Readings	2469
• Printing and Document Generation	
◦ Prerequisites	2472
◦ The Android Print System	2472
◦ About the Sample App	2473
◦ Printing a Bitmap	2474
◦ Printing an HTML Document	2476
◦ Printing a PDF File	2480
◦ Printing Using a Canvas	2488
◦ Print Jobs	2490
◦ Printing, Threads, and Services	2491
◦ Printing Prior to Android 4.4	2493
◦ HTML Generation	2494
◦ PDF Generation Options	2497
• Other System Settings and Services	
◦ Prerequisites	2499
◦ Setting Expectations	2499
◦ Can You Hear Me Now? OK, How About Now?	2504
◦ The Rest of the Gang	2507
• Dealing with Different Hardware	
◦ Prerequisites	2509
◦ Filtering Out Devices	2509
◦ Runtime Capability Detection	2512
◦ Dealing with Device Bugs	2513
• Writing and Using Parcelables	
◦ Prerequisites	2515
◦ The Role of Parcelable	2515
◦ Writing a Parcelable	2516
◦ The Limitations of Parcelable	2521
• Responding to URLs	
◦ Prerequisites	2525
◦ Manifest Modifications	2525
◦ Creating a Custom URL	2527
◦ Reacting to the Link	2528
• Plugin Patterns	
◦ Prerequisites	2531

◦ Definitions, Scenarios, and Scope	2531
◦ The Keys to Any Plugin System	2532
◦ Case Study: DashClock	2540
◦ Other Plugin Examples	2543
• PackageManager Tricks	
◦ Prerequisites	2561
◦ Asking Around	2561
◦ Preferred Activities	2565
◦ Middle Management	2570
• Searching with SearchManager	
◦ Prerequisites	2573
◦ Hunting Season	2573
◦ Search Yourself	2575
◦ Searching for Meaning In Randomness	2581
◦ May I Make a Suggestion?	2583
◦ Putting Yourself (Almost) On Par with Google	2587
• Remote Services and the Binding Pattern	
◦ Prerequisites	2593
◦ The Binding Pattern	2594
◦ When IPC Attacks!	2600
◦ Service From Afar	2602
◦ Tightening Up the Security	2608
◦ Servicing the Service	2611
◦ Thinking About Security	2615
◦ The “Everlasting Service” Anti-Pattern	2615
• Advanced Manifest Tips	
◦ Prerequisites	2617
◦ Just Looking For Some Elbow Room	2617
◦ Using an Alias	2624
◦ Getting Meta (Data)	2626
• Miscellaneous Integration Tips	
◦ Prerequisites	2631
◦ Take the Shortcut	2631
◦ Homing Beacons for Intents	2638
• Reusable Components	
◦ Prerequisites	2639
◦ Where Do I Find Them?	2639
◦ How Are They Packaged?	2640
◦ How Do I Create Them?	2641
◦ Other Considerations for Publishing Reusable Code	2644
• Android Studio Editors and Dialogs	

◦ Prerequisites	2647
◦ Project Structure	2647
◦ Translations Editor	2656
• Advanced Emulator Capabilities	
◦ Prerequisites	2663
◦ x86 Images	2663
◦ Hardware Graphics Acceleration	2666
◦ Defining New Devices	2669
◦ Keyboard Behavior	2672
◦ Headless Operation	2672
• Using Lint	
◦ Prerequisites	2673
◦ What It Is	2673
◦ When It Runs	2674
◦ What to Fix	2677
◦ What to Configure	2677
• Using Hierarchy View	
◦ Prerequisites	2683
◦ Launching Hierarchy View	2683
◦ Viewing the View Hierarchy	2684
◦ ViewServer	2687
• Using DDMS	
◦ Prerequisites	2689
◦ Starting DDMS	2689
◦ File Push and Pull	2690
◦ Screenshots	2691
◦ Location Updates	2691
◦ Placing Calls and Messages	2692
• Issues with Speed	
◦ Prerequisites	2695
◦ Getting Things Done	2695
◦ Your UI Seems... Janky	2696
◦ Not Far Enough in the Background	2696
◦ Playing with Speed	2697
• Finding CPU Bottlenecks	
◦ Prerequisites	2699
◦ Traceview	2700
◦ Other General CPU Measurement Techniques	2711
◦ UI “Jank” Measurement	2712
• Focus On: NDK	
◦ Prerequisites	2729

◦ The Role of the NDK	2730
◦ NDK Installation and Project Setup	2733
◦ Writing Your Makefile(s)	2737
◦ Building Your Library	2738
◦ Using Your Library Via JNI	2739
◦ Building and Deploying Your Project	2744
◦ Gradle and the NDK	2746
• Improving CPU Performance in Java	
◦ Prerequisites	2753
◦ Reduce CPU Utilization	2753
◦ Reduce Time on the Main Application Thread	2758
◦ Improve Throughput and Responsiveness	2766
• Finding and Eliminating Jank	
◦ Prerequisites	2769
◦ The Case: ThreePaneDemoBC	2769
◦ Are We Janky?	2770
◦ Finding the Source of the Jank	2770
◦ Where Things Went Wrong	2780
◦ Removing the Jank	2781
• Issues with Bandwidth	
◦ Prerequisites	2783
◦ You're Using Too Much of the Slow Stuff	2784
◦ You're Using Too Much of the Expensive Stuff	2784
◦ You're Using Too Much of Somebody Else's Stuff	2785
◦ You're Using Too Much... And There Is None	2786
• Focus On: TrafficStats	
◦ Prerequisites	2787
◦ TrafficStats Basics	2787
◦ Example: TrafficMonitor	2789
◦ Other Ways to Employ TrafficStats	2797
• Measuring Bandwidth Consumption	
◦ Prerequisites	2799
◦ On-Device Measurement	2799
◦ Off-Device Measurement	2801
◦ Tactical Measurement in DDMS	2803
• Being Smarter About Bandwidth	
◦ Prerequisites	2807
◦ Bandwidth Savings	2807
◦ Bandwidth Shaping	2812
◦ Avoiding Metered Connections	2816
• Issues with Application Heap	

◦ Prerequisites	2817
◦ You Are in a Heap of Trouble	2818
◦ Determining Your Heap Size At Runtime	2818
◦ Fragments of Memory	2819
◦ Getting a Trim	2820
◦ Warning: Contains Graphic Images	2821
◦ Releasing SQLite Memory	2832
◦ In Too Deep (on the Stack)	2832
• Finding Memory Leaks with MAT	
◦ Prerequisites	2835
◦ Setting Up MAT	2835
◦ Getting Heap Dumps	2836
◦ Basic MAT Operation	2842
◦ Some Leaks and Their MAT Analysis	2849
• Issues with System RAM	
◦ Prerequisites	2857
◦ Can't We All Just Get Along?	2857
◦ Contributors to System RAM Consumption	2858
◦ Measuring System RAM Consumption: Tools	2859
◦ Measuring System RAM Consumption: Runtime	2874
◦ Learn To Let Go (Of Your Heap)	2875
• Issues with Battery Life	
◦ Prerequisites	2877
◦ You're Getting Blamed	2878
◦ Not All Batteries Are Created Equal	2879
◦ Stretching Out the Last mWh	2879
• Power Measurement Options	
◦ Prerequisites	2881
◦ batterystats and the Battery Historian	2882
◦ The Qualcomm Tool (That Must Not Be Named)	2892
◦ PowerTutor	2893
◦ Battery Screen in Settings Application	2897
◦ BatteryInfo Dump	2899
• Sources of Power Drain	
◦ Prerequisites	2903
◦ Screen	2904
◦ Disk I/O	2905
◦ WiFi and Mobile Data	2906
◦ GPS	2909
◦ Camera	2910
◦ Additional Sources	2910

• Addressing Application Size Issues	
◦ Prerequisites	2913
◦ Java Code, and the 64K Method Limit	2913
◦ Native Code	2917
◦ Images	2919
◦ APK Expansion Files	2921
• The Role of Scripting Languages	
◦ Prerequisites	2923
◦ All Grown Up	2923
◦ Following the Script	2924
◦ Going Off-Script	2925
• The Scripting Layer for Android	
◦ Prerequisites	2929
◦ The Role of SL4A	2929
◦ Getting Started with SL4A	2930
◦ Writing SL4A Scripts	2938
◦ Running SL4A Scripts	2943
◦ Potential Issues	2946
• JVM Scripting Languages	
◦ Prerequisites	2949
◦ Languages on Languages	2949
◦ A Brief History of JVM Scripting	2950
◦ Limitations	2951
◦ SL4A and JVM Languages	2952
◦ Embedding JVM Languages	2952
◦ Other JVM Scripting Languages	2965
• The Role of Alternative Environments	
◦ Prerequisites	2969
◦ In the Beginning, There Was Java... ..	2970
◦ ... And It Was OK	2970
◦ Bucking the Trend	2971
◦ Support, Structure	2971
◦ Caveat Developer	2972
• HTML5	
◦ Prerequisites	2973
◦ Offline Applications	2973
◦ Web Storage	2980
◦ Going To Production	2983
◦ Issues You May Encounter	2984
◦ HTML5: The Baseline	2987
• PhoneGap	

◦ Prerequisites	2989
◦ What Is PhoneGap?	2989
◦ Using PhoneGap	2992
◦ PhoneGap and the Checklist Sample	2997
◦ Issues You May Encounter	3002
◦ For More Information	3005
• Other Alternative Environments	
◦ Prerequisites	3007
◦ Rhodes	3007
◦ Flash, Flex, and AIR	3008
◦ JRuby and Ruboto	3008
◦ App Inventor	3009
◦ Titanium Mobile	3011
◦ Other JVM Compiled Languages	3012
• In-App Diagnostics	
◦ Prerequisites	3015
◦ The Diagnostic Activity	3016
◦ The Diagnostic Web App	3025
◦ The Diagnostic Overlay	3034
• Anti-Patterns	
◦ Prerequisites	3043
◦ Leak Threads... Or Things Attached to Threads	3043
◦ Use Large Heap Unnecessarily	3045
◦ Misuse the MENU Button	3047
◦ Interfere with Navigation	3048
◦ Use <code>android:sharedUserId</code>	3050
◦ Implement a “Quit” Button	3051
◦ Terminate Your Process	3053
◦ Try to Hide from the User	3054
◦ Use Multiple Processes	3055
◦ Hog System Resources	3057
• Widget Catalog: AdapterViewFlipper	
◦ Key Usage Tips	3059
◦ A Sample Usage	3060
◦ Visual Representation	3060
• Widget Catalog: CalendarView	
◦ Key Usage Tips	3061
◦ A Sample Usage	3062
◦ Visual Representation	3062
• Widget Catalog: DatePicker	
◦ Key Usage Tips	3065

◦ A Sample Usage	3066
◦ Visual Representation	3067
• Widget Catalog: ExpandableListView	
◦ Key Usage Tips	3071
◦ A Sample Usage	3072
◦ Visual Representation	3077
• Widget Catalog: SeekBar	
◦ Key Usage Tips	3081
◦ A Sample Usage	3081
◦ Visual Representation	3083
• Widget Catalog: SlidingDrawer	
◦ Key Usage Tips	3085
◦ A Sample Usage	3086
◦ Visual Representation	3087
• Widget Catalog: StackView	
◦ Key Usage Tips	3089
◦ A Sample Usage	3090
◦ Visual Representation	3091
• Widget Catalog: TabHost and TabWidget	
◦ Deprecation Notes	3093
◦ Key Usage Tips	3093
◦ A Sample Usage	3094
◦ Visual Representation	3096
• Widget Catalog: TimePicker	
◦ Key Usage Tips	3099
◦ A Sample Usage	3099
◦ Visual Representation	3101
• Widget Catalog: ViewFlipper	
◦ Key Usage Tips	3103
◦ A Sample Usage	3104
◦ Visual Representation	3105
• Chrome and Chrome OS	
◦ Prerequisites	3108
◦ Welcome to the ARC	3108
◦ The ARC Welder	3108
◦ Packaging Your Android App for Chrome	3110
◦ Distribution Options	3115
◦ Trying Your App	3118
◦ Your App on Chrome OS	3123
◦ Apps Sans Role	3134
◦ Other Security Notes	3134

◦ Getting Help	3134
• Device Catalog: Kindle Fire	
◦ Prerequisites	3137
◦ Introducing the Kindle Fire series	3137
◦ What Features and Configurations Does It Use?	3138
◦ What Is Really Different?	3140
◦ Getting Your Development Environment Established	3146
◦ How Does Distribution Work?	3148
◦ Amazon Equivalents of Google Services	3149
◦ Getting Help with the Kindle Fire	3150
• Device Catalog: BlackBerry	
◦ <i>I Thought BlackBerry Had Their Own OS?</i>	3151
◦ <i>What Else Is Different?</i>	3152
◦ <i>What Are We Making?</i>	3154
◦ <i>Getting Your Development Environment Established</i>	3155
◦ <i>How Does Distribution Work?</i>	3157
• Device Catalog: Wrist Wearables	
◦ Prerequisites	3160
◦ Divvying Up the Wearables Space	3160
◦ Example Wrist Wearables	3161
◦ Strategic Considerations	3164
◦ Tactical Considerations	3166
◦ What About Android Wear?	3169
• Device Catalog: Android TV	
◦ Prerequisites	3171
◦ Hey, Wait a Minute... I Thought the Name Was “Google TV”?	3171
◦ Some Android TV Hardware	3172
◦ What Features and Configurations Does It Use?	3174
◦ What Is Really Different?	3175
◦ Getting Your Development Environment Established	3177
◦ How Does Distribution Work?	3179
• Device Catalog: Amazon Fire TV and Fire TV Stick	
◦ Prerequisites	3183
◦ Introducing the Fire TV Devices	3183
◦ What Features and Configurations Do They Use?	3189
◦ What Is Really Different?	3191
◦ <i>Casting and Fire TV</i>	3192
◦ Getting Your Development Environment Established	3193
◦ Working with the Remote and Controller	3195
◦ How Does Distribution Work?	3197
◦ Getting Help	3197

- [Appendix A: CWAC Libraries](#)
 - cwac-adapter 3199
 - cwac-camera 3199
 - cwac-colormixer 3200
 - cwac-layouts 3200
 - cwac-mediarouter 3200
 - cwac-merge 3200
 - cwac-pager 3201
 - cwac-presentation 3201
 - cwac-provider 3201
 - cwac-richedit 3201
 - cwac-richtextutils 3202
 - cwac-sacklist 3202
 - cwac-security 3202
 - cwac-strictmodeex 3202
 - cwac-wakeful 3202
- [Appendix B: M Developer Preview](#)
 - *What Is the M Developer Preview?* 3204
 - *Getting and Using the M Developer Preview* 3204
 - *What's New in the M Developer Preview?* 3207
 - *New Support Libraries* 3235
 - *Supporting MNC and Older Devices* 3237

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 2,800+ 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

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

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](#)
- [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](#)

Security

- [SSL](#)
- [Encrypted Storage](#)
- [Advanced Permissions](#)
- [Restricted Profiles and UserManager](#)
- [Tapjacking](#)
- [Miscellaneous Security Techniques](#)

Hardware and System Services

- [Accessing Location-Based Services](#)
- [The Fused Location Provider](#)
- [Working with the Clipboard](#)
- [Telephony](#)
- [Working With SMS](#)
- [NFC](#)
- [Device Administration](#)
- [PowerManager and WakeLocks](#)
- [JobScheduler](#)
- [Push Notifications with GCM](#)
- [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: M Developer Preview](#)

About the Updates

This book is updated frequently, typically every 6-8 weeks.

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.

PREFACE

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 [commonsguy](#) 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.
- 80% off of live webinars hosted by Mark Murphy, the author of this book, on Android application development topics.

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 Eclipse and Android Studio.

If you are using Eclipse, please do **NOT** import all of the projects from the repo into your main workspace. There are *hundreds* of these projects, and they may cause your Eclipse environment to become very slow, particularly when starting it up. Instead, import only those specific projects that you want to work with “live” as opposed to simply reading about them in the book.

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 June 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.

PREFACE

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.

Portions of this book are reproduced from work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.

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.

KEY ANDROID CONCEPTS



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:

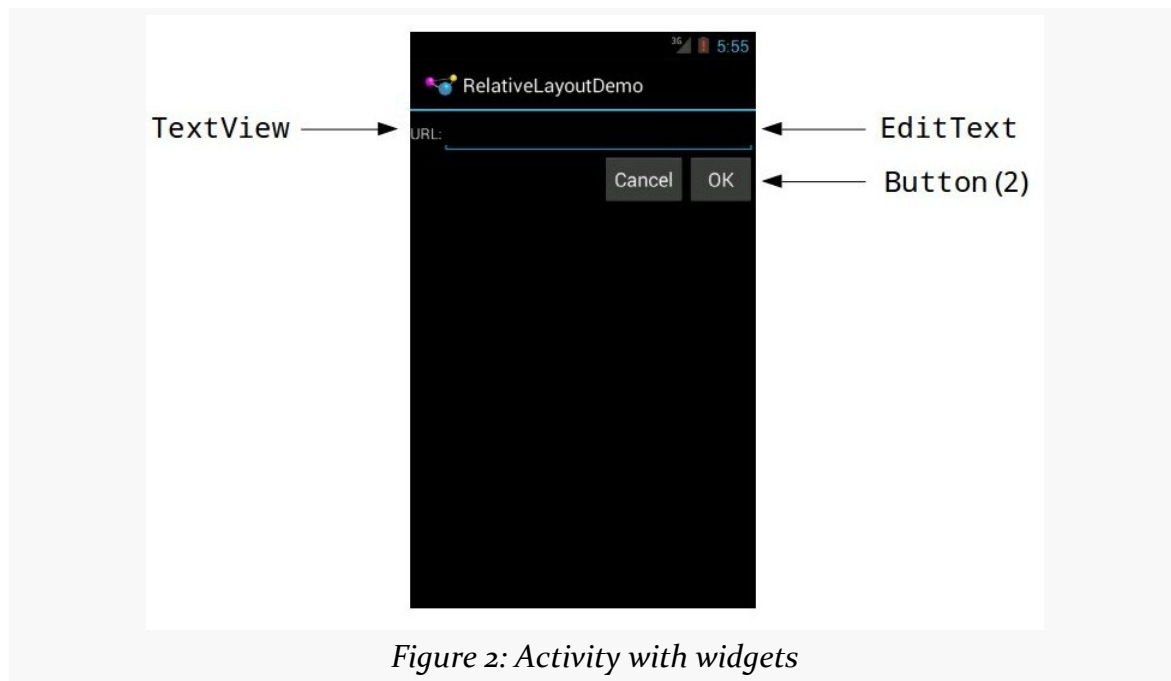


Figure 2: Activity with widgets

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.

KEY ANDROID CONCEPTS

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 15 (Android 4.0.3)
- API Level 16 (Android 4.1)
- API Level 17 (Android 4.2)
- API Level 18 (Android 4.3)
- API Level 19 (Android 4.4)
- API Level 21 (Android 5.0)

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.

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.

For historical reasons, the [next chapter contains a section with instructions](#) on how to set up Eclipse for Android development, as part of getting an overall Android development environment established. Similarly, there will be Eclipse instructions in the tutorials and notes about using Eclipse elsewhere in the book. These will be removed from a future edition of the book, or perhaps migrated over to Andmore.

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 Gradle for Android plugin) 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 lib32bz2-1.0 lib32stdc++6
```

Step #3 - Install the Developer Tools

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

There are two major options that this book covers:

1. Use [Android Studio](#)
2. Use Eclipse, [adding in the ADT plugin to an existing Eclipse installation](#)

A third option would be to skip either IDE and just use the Android SDK and Gradle for Android directly. This would only be appropriate for fairly expert developers, and so the tutorials do not walk you through this path. This book has [several chapters on Gradle](#), but those chapters are presently designed for developers already experienced

TUTORIAL #1 - INSTALLING THE TOOLS

in working with Android apps. For the purposes of learning Android, you are better served working with one of the supported IDEs initially. Later on, if you wish to drop the IDE and use Gradle for Android directly, you are welcome to do so.

Note that you will also hear reference to an “ADT Bundle” that Google used to distribute, consisting of a copy of Eclipse with the ADT plugin pre-installed. This is no longer available for download.

As noted earlier in the book, Eclipse support has been officially dropped by Google and, therefore, you **really** should consider using Android Studio over Eclipse today. Otherwise, you are welcome to choose either of these options. The installation instructions for each are described in the sections that follow.

Option #1 - Android Studio

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 1.2.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.

Option #2 - Install the ADT for Eclipse

To use Eclipse, you will need to install the base Android SDK, then configure Eclipse.

Install the Android SDK

The Android developer tools can be found on the [Android Developers Web site](#).

You will want to click on “Using an Existing IDE” (even if you have not yet installed Eclipse) and download the ZIP or TGZ file presented to you, unpacking it in some likely spot — there is no specific path that is required. Windows users also have the option of running a self-installing EXE file.

TUTORIAL #1 - INSTALLING THE TOOLS

Configure Eclipse

If you have not yet installed Eclipse, you will need to do that first. Eclipse can be downloaded from the [Eclipse Web site](http://eclipse.org). The “Eclipse IDE for Java Developers” package will work fine. Note that the Android tools require Eclipse 3.7 (Indigo) or newer at the time of this writing.

If you already had Eclipse installed, it is a good idea for you to go in and check your compiler compliance level (Preferences > Java > Compiler). That should be set to 1.6. Notably, this allows the use of `@Override` annotations to indicate methods that are implementing a Java interface, rather than truly overriding a superclass method. This annotation is very common in Java code in Android projects (including many of the samples in this book).

Next, you need to install the Android Developer Tools (ADT) plug-in. To do this, go to Help | Install New Software... in the Eclipse main menu. Then, click the Add button to add a new source of plug-ins. Give it some name (e.g., Android) and supply the following URL: <https://dl-ssl.google.com/android/eclipse/>. That should trigger Eclipse to download the roster of plug-ins available from that site:

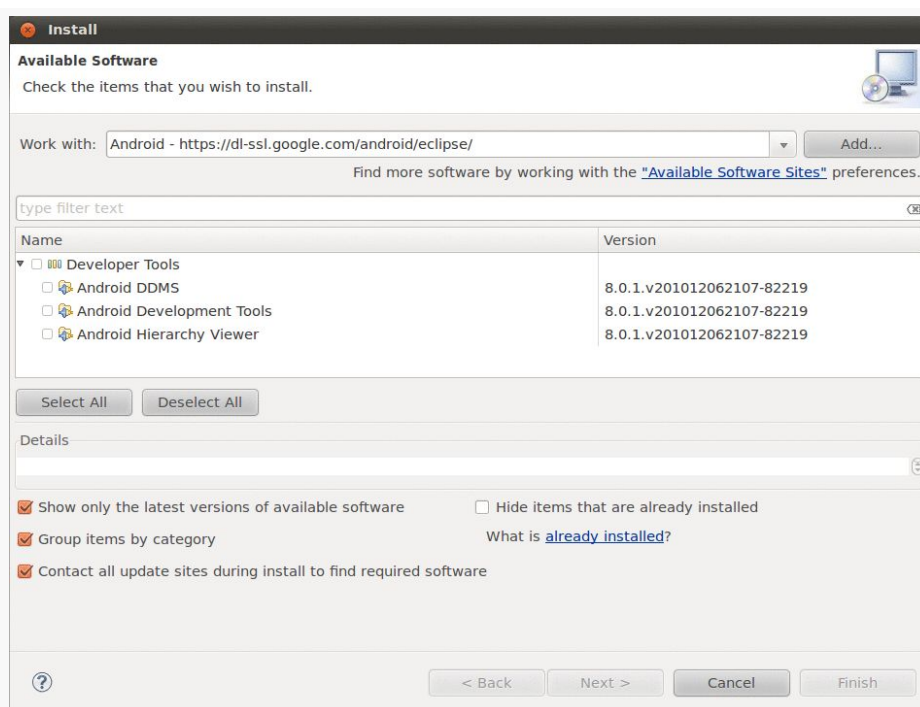
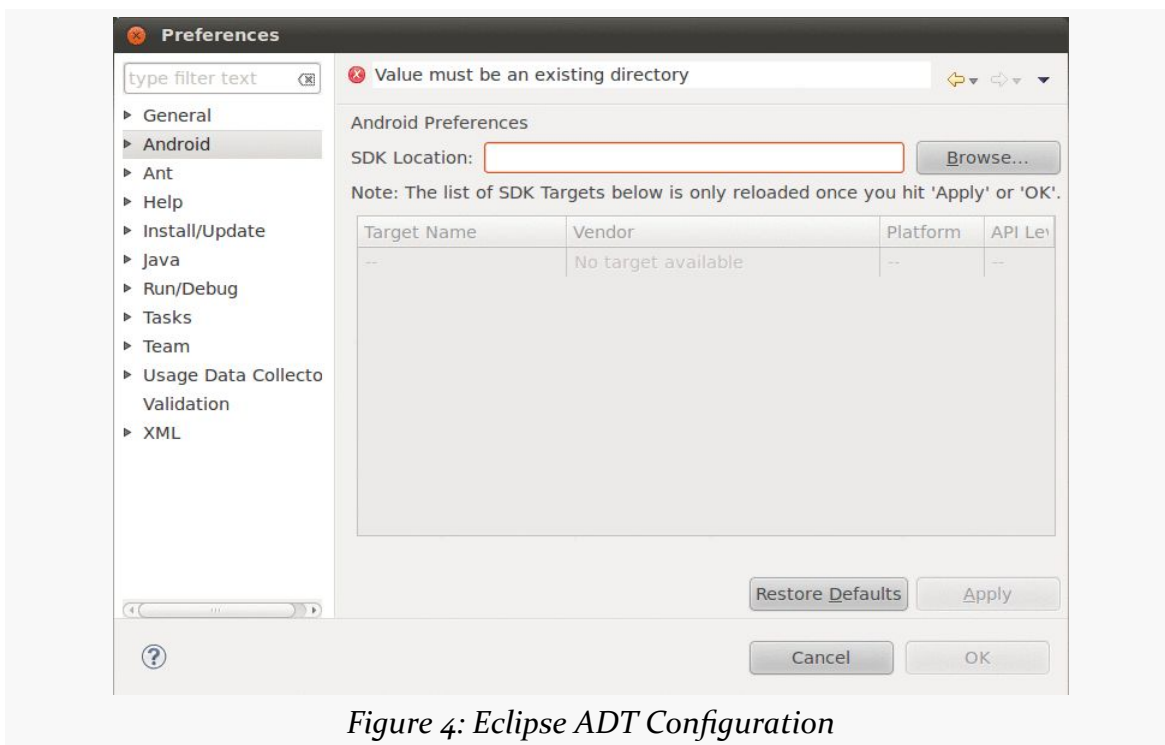


Figure 3: Eclipse ADT plug-in installation

TUTORIAL #1 - INSTALLING THE TOOLS

Check the checkbox to the left of “Developer Tools” and click the Next button. Follow the rest of the wizard to review the tools to be downloaded and their respective license agreements. When the Finish button is enabled, click it, and Eclipse will download and install the plug-ins. When done, Eclipse will ask to restart — please let it.

Then, you need to teach ADT where your Android SDK installation is from [the preceding section](#). This should occur on your next restart of Eclipse, via a “welcome wizard”. Otherwise, to do this, choose Window | Preferences from the Eclipse main menu (or the equivalent Preferences option for OS X). Click on the Android entry in the list on the left:



Then, click the Browse... button to find the directory where you installed the SDK. After choosing it, click Apply on the Preferences window, and you should see the Android SDK versions you installed previously. Then, click OK, and the ADT will be ready for use.

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.

Android Studio First Run and SDK Manager Launch

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

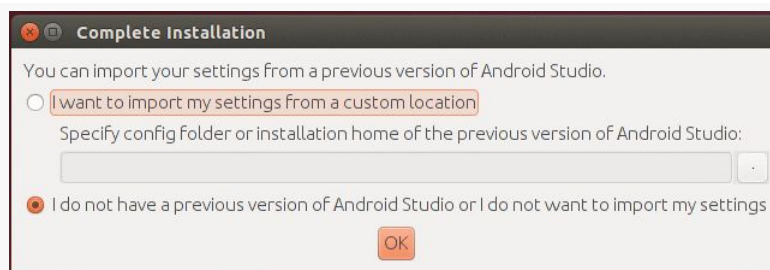


Figure 5: Android Studio First-Run Settings Migration Dialog

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:

TUTORIAL #1 - INSTALLING THE TOOLS

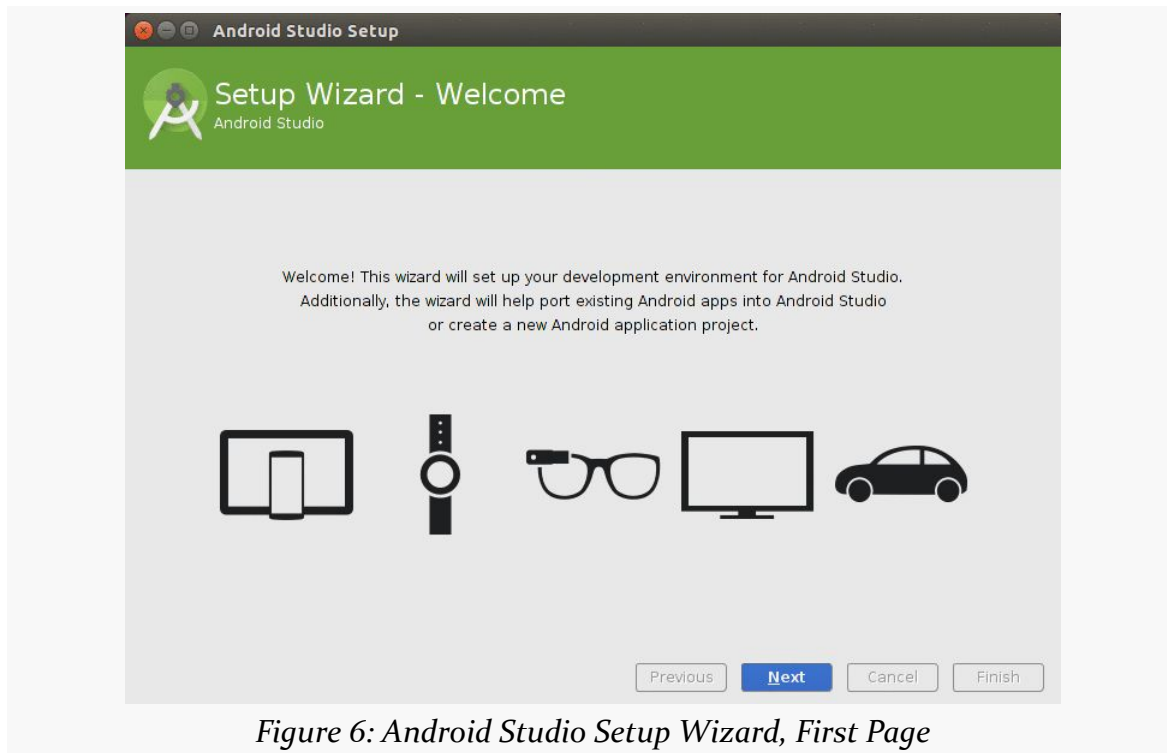


Figure 6: Android Studio Setup Wizard, First Page

Just click “Next” to advance to the second page of the wizard:

TUTORIAL #1 - INSTALLING THE TOOLS

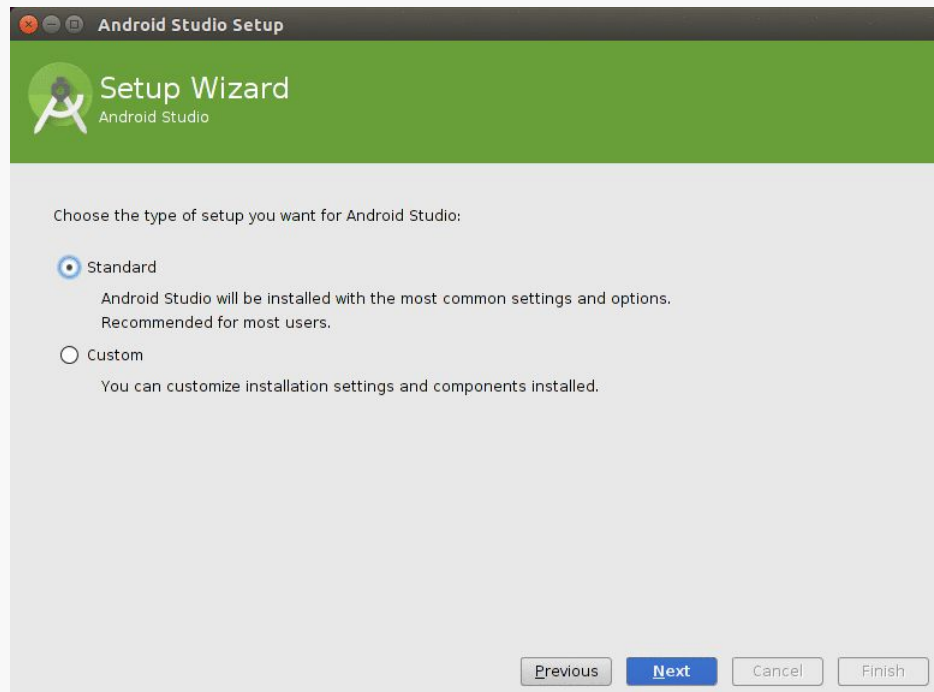


Figure 7: Android Studio Setup Wizard, Second Page

Here, you have a choice between “Standard” and “Custom” setup modes. “Custom” simply allows you to indicate what should be set up:

TUTORIAL #1 - INSTALLING THE TOOLS

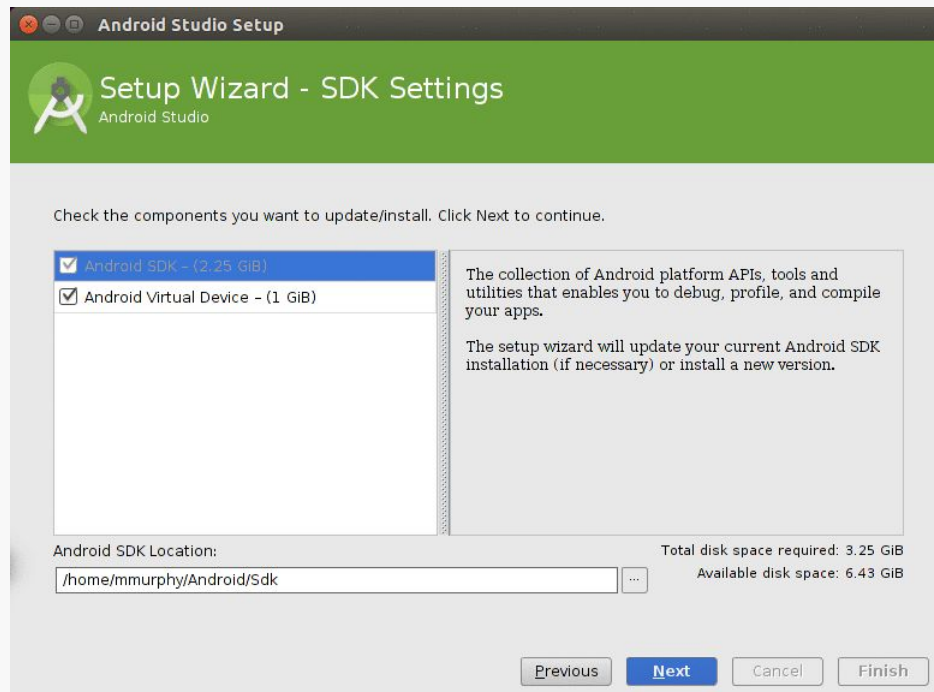


Figure 8: Android Studio Setup Wizard, Custom Page

Most likely, right now, you need all of that anyway, so the “Standard” route would be fine.

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

TUTORIAL #1 - INSTALLING THE TOOLS

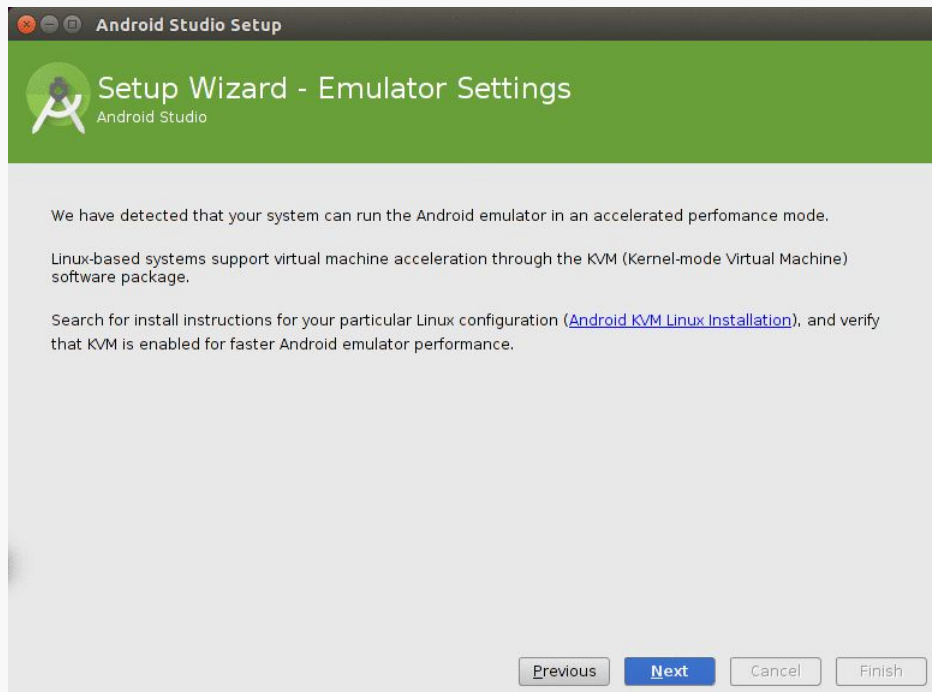


Figure 9: Android Studio Setup Wizard, Emulator Info Page

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 need to review the various licenses and click “Accept” if you wish to continue. 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, try installing the following Linux packages: `lib32z1 lib32ncurses5 lib32bz2-1.0 lib32stdc++6`.

When that is done, you will be taken to the Android Studio Welcome dialog:

TUTORIAL #1 - INSTALLING THE TOOLS

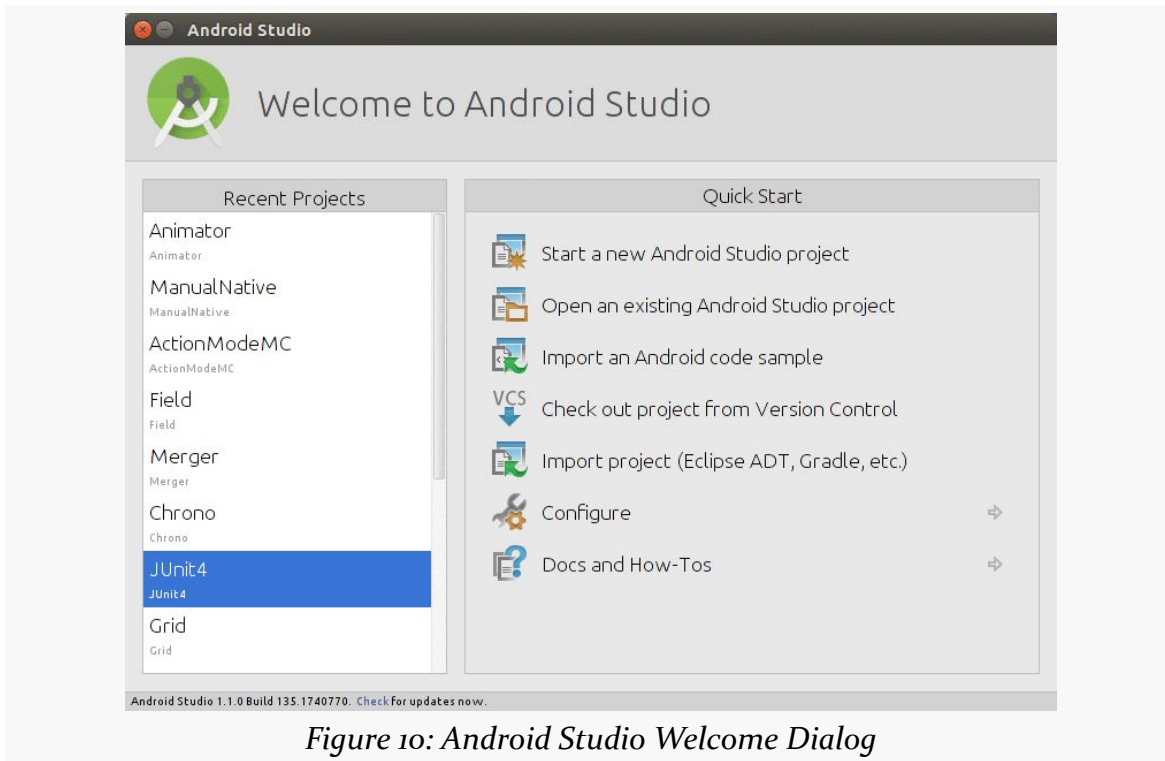


Figure 10: Android Studio Welcome Dialog

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:

TUTORIAL #1 - INSTALLING THE TOOLS

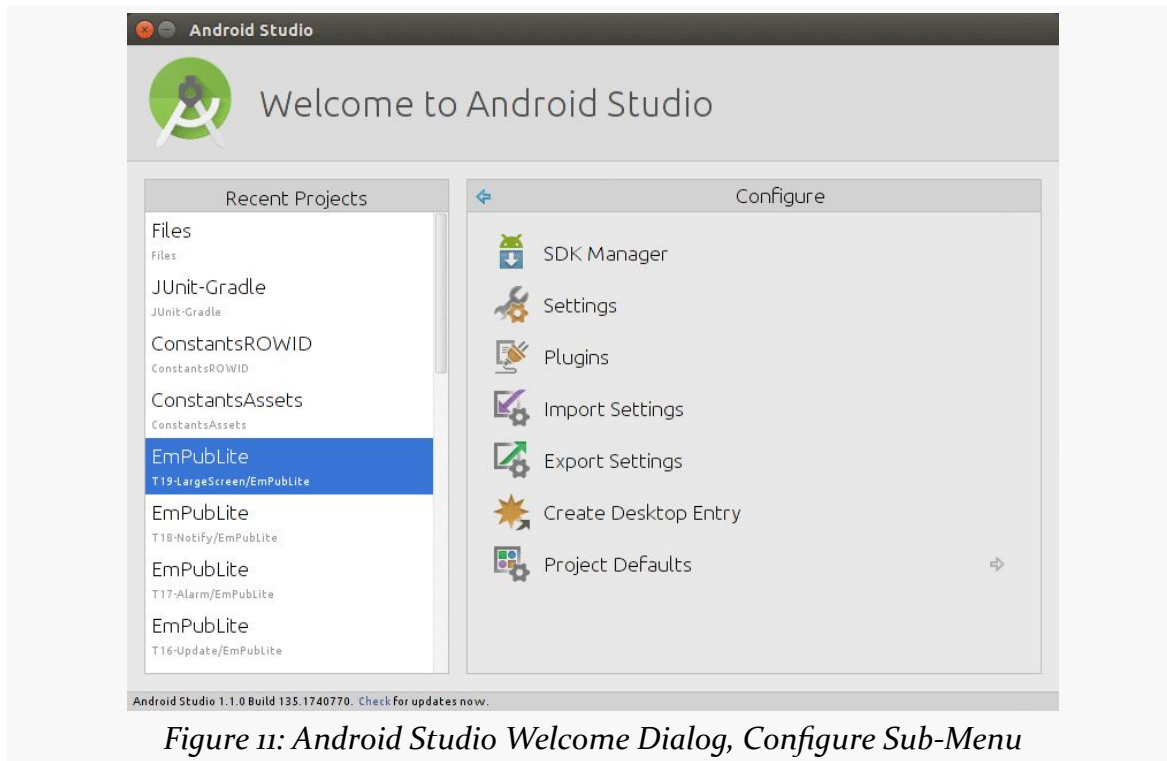


Figure 11: Android Studio Welcome Dialog, Configure Sub-Menu

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

Eclipse SDK Manager Launch

Eclipse may take a few extra moments on its first launch to get going, but it does not do anything especially unusual.

Then, choose Windows > Android SDK Manager from the main menu, or tap on the toolbar icon that looks like the Android “bugdroid” mascot peeking out of the top of a box with a downward-pointing white arrow.

Using SDK Manager and Updating Your Environment

You should now have the SDK Manager window open:

TUTORIAL #1 - INSTALLING THE TOOLS

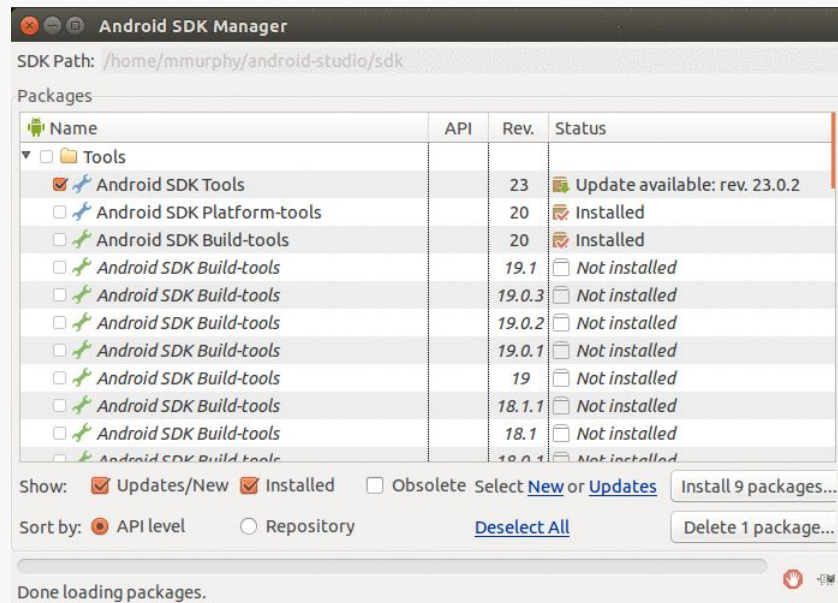


Figure 12: Android SDK Manager

At this point, while you have some of the build tools, you may lack the Java files necessary to compile an Android application. You also lack a few additional build tools, plus the files necessary to run an Android emulator. The checkboxes indicate which packages you want to install — by default, it pre-checks a number of them. If you chose the “ADT Bundle”, some things will already be pre-installed for you.

You will want to install the following items, if they have not already been installed:

1. Android SDK Tools, Platform-tools, and the latest **production** Build-tools.
2. “SDK Platform” for all Android SDK releases you want to code against — for this version of this book API 19 (Android 4.4) is recommended, along with any others with which you wish to experiment.
3. “ARM EABI v7a System Image”, if there is an option for that for whatever API level(s) you want to test against. You can also download the “Intel x86 Atom System Image”, if one is available to you, as it is much faster, though setting that up is [a bit of an advanced topic](#). You are also welcome to download similar images for any other Android API level that you are interested in testing against. However, for the purposes of this book, **DO NOT** choose the Android 4.4 or 4.4W emulator images, as the Android 4.4 emulator has a bug and the Android 4.4W emulator is only for Android Wear devices.
4. “Documentation for Android SDK” for the latest Android SDK release.

TUTORIAL #1 - INSTALLING THE TOOLS

5. “Samples for SDK” for the Android SDK release you chose in item #2 above, and perhaps for older releases if you wish.
6. Android Support Library and the Android Support Repository (in the Extras group at the bottom of the tree).

If you are running Windows, also choose the Google USB Driver (in the Extras group at the bottom of the tree).

Also, if anything that you presently have installed has updates available, they should already be pre-checked and will be updated when you install the items that you are adding.

Conversely, if anything labeled “preview” is checked, uncheck 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.

Then, click the Install button beneath the tree on the right, which brings up a license confirmation dialog:

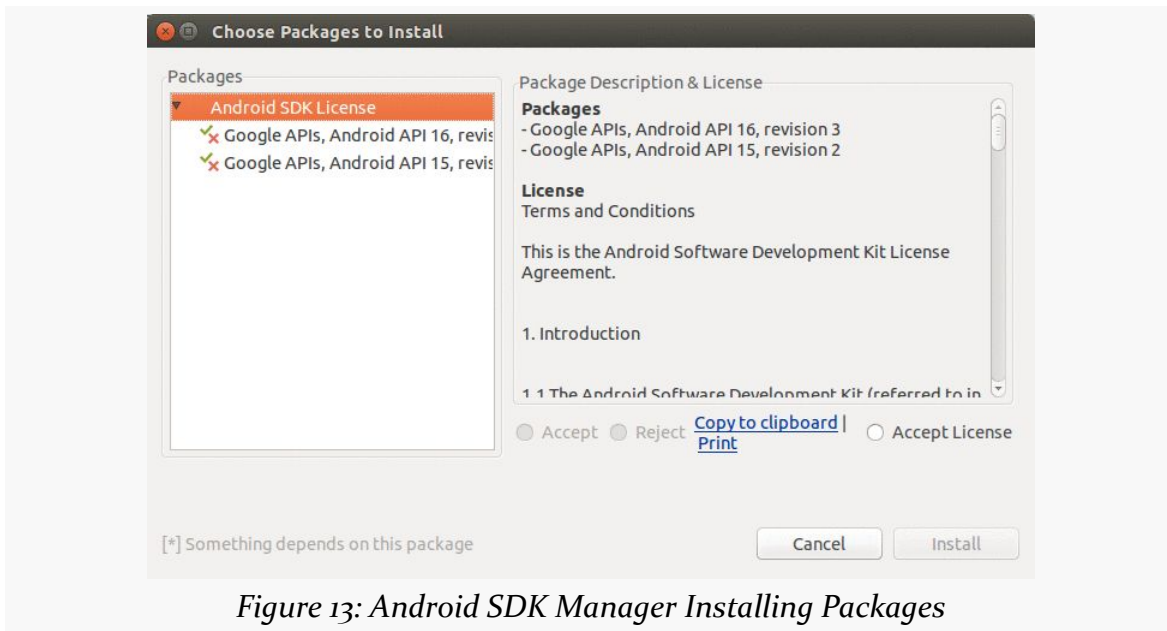


Figure 13: Android SDK Manager Installing Packages

Review and accept the licenses, then click the Install button.

When the download is complete, you can close up the SDK Manager.

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.