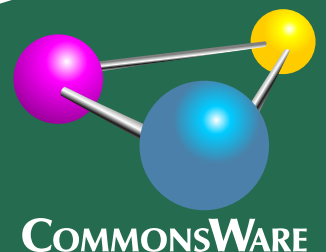


**Version
6.3**

*Supports Through
Android 5.0!*

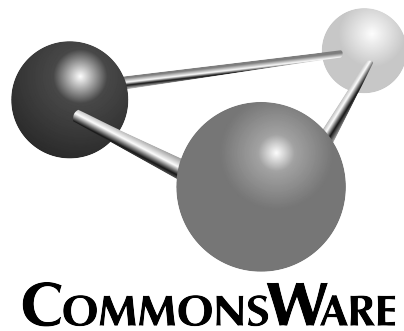
The Busy Coder's Guide to Android Development

Mark L. Murphy



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 © 2008-2014 CommonsWare, LLC. All Rights Reserved.
Printed in the United States of America.

Printing History:
December 2014: Version 6.3 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!	xxxv
◦ The Book's Structure	xxxv
◦ <i>The Trails</i>	xxxvi
◦ About the Updates	xli
◦ Warescription	xli
◦ Book Bug Bounty	xlii
◦ Source Code and Its License	xliii
◦ <i>Creative Commons and the Four-to-Free (42F) Guarantee</i>	xliv
◦ Acknowledgments	xliv
• Key Android Concepts	
◦ Android Applications	1
◦ <i>Android Devices</i>	7
◦ Don't Be Scared	10
• Choosing Your Development Toolchain	
◦ <i>Android Studio</i>	11
◦ <i>Eclipse</i>	11
◦ IntelliJ IDEA	12
◦ Command-Line Builds via Gradle for Android	12
◦ Yet Other Alternatives	13
◦ IDEs... And This Book	13
◦ <i>What We Are Not Covering</i>	13
• Tutorial #1 - Installing the Tools	
◦ <i>Step #1 - Checking Your Hardware Requirements</i>	15
◦ <i>Step #2 - Setting Up Java and 32-Bit Linux Support</i>	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
◦ <i>Projects and Android Studio</i>	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
• 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
• Inside the Manifest	
◦ An Application For Your Application	95
◦ Specifying Versions	96
◦ Supporting Multiple Screens	96
◦ Other Stuff	97
• Introducing Gradle	
◦ The Big Questions	99
◦ Obtaining Gradle	102
◦ Versions of Gradle and Gradle for Android	104
◦ Gradle Environment Variables	105
◦ Examining the Gradle File	105
◦ Learning More About Gradle	109
◦ Visit the Trails!	109
• 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	128
◦ The Resource That Shall Not Be Named... Yet	130

• Tutorial #4 - Adjusting Our Resources	
◦ Step #1: Changing the Name	131
◦ Step #2: Changing the Icon	133
◦ Step #3: Running the Result	142
◦ In Our Next Episode...	144
• The Theory of Widgets	
◦ What Are Widgets?	145
◦ Size, Margins, and Padding	147
◦ What Are Containers?	147
◦ The Absolute Positioning Anti-Pattern	148
◦ <i>The Theme of This Section: Themes</i>	149
• The Android User Interface	
◦ The Activity	153
◦ Dissecting the Activity	154
◦ Using XML-Based Layouts	155
• Basic Widgets	
◦ Common Concepts	161
◦ <i>Assigning Labels</i>	164
◦ A Commanding Button	174
◦ Fleeting Images	179
◦ Fields of Green. Or Other Colors.	186
◦ <i>More Common Concepts</i>	192
◦ Visit the Trails!	194
• Debugging Your App	
◦ Get Thee To a Stack Trace	196
◦ The Case of the Confounding Class Cast	200
◦ Point Break	201
• LinearLayout and the Box Model	
◦ Concepts and Properties	203
◦ <i>Android Studio Graphical Layout Editor</i>	207
◦ Eclipse Graphical Layout Editor	208
• Other Common Widgets and Containers	
◦ Just a Box to Check	211
◦ Don't Like Checkboxes? How About Toggles or Switches?	216
◦ Turn the Radio Up	221
◦ All Things Are Relative	225
◦ Tabula Rasa	232
◦ Scrollwork	239
◦ Making Progress with ProgressBars	242
◦ Visit the Trails!	243
• Tutorial #5 - Making Progress	

◦ Step #1: Removing The “Hello, World”	245
◦ Step #2: Adding a ProgressBar	247
◦ Step #3: Seeing the Results	250
◦ In Our Next Episode...	250
• GUI Building, Continued	
◦ Making Your Selection	251
◦ Including Includes	252
◦ Wrap It Up (In a Container)	254
◦ Morphing Widgets	255
◦ Preview of Coming Attractions	256
• AdapterViews and Adapters	
◦ Adapting to the Circumstances	257
◦ Lists of Naughty and Nice	259
◦ Clicks versus Selections	261
◦ Spin Control	265
◦ Grid Your Lions (Or Something Like That...)	268
◦ Fields: Now With 35% Less Typing!	271
◦ Customizing the Adapter	275
◦ Visit the Trails!	283
• The WebView Widget	
◦ Role of WebView	285
◦ WebView and WebKit	286
◦ Adding the Widget	286
◦ Loading Content Via a URL	287
◦ Supporting JavaScript	288
◦ Alternatives for Loading Content	289
◦ Listening for Events	290
◦ WebView and Android 4.4	293
◦ Visit the Trails!	294
• Defining and Using Styles	
◦ Styles: DIY DRY	295
◦ Elements of Style	297
◦ Themes: Would a Style By Any Other Name...	300
◦ What Happens If You Have No Theme	300
• JARs and Library Projects	
◦ The Dalvik VM, and a Bit of ART	304
◦ Getting the Library	304
◦ The Outer Limits	306
◦ JAR Dependency Management	307
◦ OK, So What is a Library Project?	307
◦ Using a Library Project	308

◦ Library Projects: What You Get	309
◦ <i>The Android Support Package</i>	310
• Tutorial #6 - Adding a Library	
◦ <i>Step #1: Attaching the Android Support Package</i>	315
◦ <i>Step #2: Downloading the Third-Party JARs</i>	316
◦ In Our Next Episode...	318
• The Action Bar	
◦ <i>Bar Hopping</i>	319
◦ Yet Another History Lesson	325
◦ <i>Your Action Bar Options</i>	326
◦ <i>Setting the Target</i>	327
◦ <i>Defining the Resource</i>	329
◦ <i>Applying the Resource</i>	331
◦ <i>Responding to Events</i>	331
◦ <i>The Rest of the Sample Activity</i>	332
◦ MENU Key, We Hardly Knew Ye	338
◦ <i>Action Bars, Live in Living Color!</i>	339
◦ Visit the Trails!	349
• Tutorial #7 - Setting Up the Action Bar	
◦ Step #1: Acquiring Some Icons	351
◦ Step #2: Defining Some Options	352
◦ <i>Step #3: Loading and Responding to Our Options</i>	355
◦ In Our Next Episode...	359
• Android's Process Model	
◦ When Processes Are Created	361
◦ BACK, HOME, and Your Process	362
◦ Termination	363
◦ Foreground Means "I Love You"	364
◦ You and Your Heap	364
• Activities and Their Lifecycles	
◦ <i>Creating Your Second (and Third and...) Activity</i>	366
◦ Warning! Contains Explicit Intents!	372
◦ Using Implicit Intents	374
◦ Extra! Extra!	379
◦ Pondering Parcelable	381
◦ Asynchronicity and Results	382
◦ Schroedinger's Activity	382
◦ Life, Death, and Your Activity	383
◦ When Activities Die	385
◦ Walking Through the Lifecycle	386
◦ Recycling Activities	389

◦ Application: Transcending the Activity	390
• Tutorial #8 - Setting Up An Activity	
◦ Step #1: Creating the Stub Activity Class and Manifest Entry	393
◦ Step #2: Launching Our Activity	396
◦ In Our Next Episode...	397
• The Tactics of Fragments	
◦ The Six Questions	399
◦ Where You Get Your Fragments From	402
◦ Your First Fragment	402
◦ The Fragment Lifecycle Methods	405
◦ Your First Dynamic Fragment	406
◦ Fragments and the Action Bar	409
◦ Fragments Within Fragments: Just Say “Maybe”	411
◦ Fragments and Multiple Activities	411
• Tutorial #9 - Starting Our Fragments	
◦ Step #1: Create a SimpleContentFragment	413
◦ Step #2: Examining SimpleContentFragment	416
◦ In Our Next Episode...	416
• Swiping with ViewPager	
◦ Swiping Design Patterns	417
◦ Pieces of a Pager	418
◦ Paging Fragments	418
◦ Paging Other Stuff	423
◦ Indicators	424
• Tutorial #10 - Rigging Up a ViewPager	
◦ Step #1: Add a ViewPager to the Layout	429
◦ Step #2: Obtaining Our ViewPager	430
◦ Step #3: Creating a ContentsAdapter	431
◦ Step #4: Setting Up the ViewPager	432
◦ In Our Next Episode...	433
• Resource Sets and Configurations	
◦ What’s a Configuration? And How Do They Change?	435
◦ Configurations and Resource Sets	436
◦ Screen Size and Orientation	437
◦ Coping with Complexity	440
◦ Choosing The Right Resource	441
◦ API-Versioned Resources	445
◦ Default Change Behavior	447
◦ State Saving Scenarios	448
◦ Your Options for Configuration Changes	450
◦ Blocking Rotations	461

◦ <i>And Now, a Word From the Android Project View</i>	461
• Material Design Basics	
◦ <i>Your App, in Technicolor!</i>	465
• Dealing with Threads	
◦ The Main Application Thread	473
◦ Getting to the Background	475
◦ Asyncing Feeling	475
◦ Alternatives to AsyncTask	485
◦ And Now, The Caveats	486
◦ Event Buses	487
◦ Visit the Trails!	494
• Requesting Permissions	
◦ Mother, May I?	496
◦ New Permissions in Old Applications	498
◦ Permissions: Up Front Or Not At All	498
◦ Signature Permissions	499
◦ Requiring Permissions	500
• Assets, Files, and Data Parsing	
◦ Packaging Files with Your App	501
◦ Files and Android	503
◦ Working with Internal Storage	504
◦ Working with External Storage	507
◦ Multiple User Accounts	511
◦ Linux Filesystems: You Sync, You Win	512
◦ StrictMode: Avoiding Janky Code	513
◦ XML Parsing Options	516
◦ JSON Parsing Options	517
◦ Visit the Trails!	517
• Tutorial #11 - Adding Simple Content	
◦ Step #1: Adding Some Content	519
◦ Step #2: Using SimpleContentFragment	520
◦ Step #3: Launching Our Activities, For Real This Time	521
◦ In Our Next Episode...	523
• Tutorial #12 - Displaying the Book	
◦ Step #1: Adding a Book	525
◦ Step #2: Creating a ModelFragment	526
◦ Step #3: Defining Our Model	527
◦ Step #4: Examining Our Model	529
◦ Step #5: Defining Our Event	529
◦ Step #6: Loading Our Model	531
◦ Step #7: Registering for Events	533

◦ Step #8: Adapting the Content	534
◦ Step #9: Showing the Content When Loaded	535
◦ Step #10: Attaching our ModelFragment	535
◦ Step #11: Showing the Content After a Configuration Change	537
◦ Step #12: Going Home, Again	537
◦ Step #13: Setting Up StrictMode	538
◦ In Our Next Episode...	539
• Using Preferences	
◦ Getting What You Want	541
◦ Stating Your Preference	542
◦ Introducing PreferenceActivity	543
◦ Types of Preferences	553
◦ Intents for Headers or Preferences	556
◦ Conditional Headers	557
• Tutorial #13 - Using Some Preferences	
◦ Step #1: Defining the Preference XML Files	563
◦ Step #2: Creating Our PreferenceActivity and PreferenceFragment	566
◦ Step #3: Adding To Our Action Bar	568
◦ Step #4: Launching the PreferenceActivity	569
◦ Step #5: Loading the Preferences	573
◦ Step #6: Saving the Last-Read Position	576
◦ Step #7: Restoring the Last-Read Position	578
◦ Step #9: Keeping the Screen On	578
◦ In Our Next Episode...	580
• SQLite Databases	
◦ Introducing SQLite	581
◦ Thinking About Schemas	582
◦ Start with a Helper	582
◦ Getting Data Out	587
◦ The Rest of the CRUD	593
◦ Hey, What About Hibernate?	598
◦ Visit the Trails!	598
• Tutorial #14 - Saving Notes	
◦ Step #1: Adding a DatabaseHelper	599
◦ Step #2: Examining DatabaseHelper	601
◦ Step #3: Creating a NoteFragment	601
◦ Step #4: Examining NoteFragment	606
◦ Step #5: Creating the NoteActivity	606
◦ Step #6: Examining NoteActivity	608
◦ Step #7: Add Notes to the Action Bar	608

◦ Step #8: Defining a NoteLoadedEvent	610
◦ Step #9: Loading a Note from the Database	611
◦ Step #10: Loading the Note Into the Fragment	612
◦ Step #11: Updating the Database	613
◦ Step #12: Saving the Note	614
◦ Step #13: Adding a Delete Action Bar Item	616
◦ Step #14: Closing the NoteFragment When Deleted	618
◦ In Our Next Episode...	623
• Internet Access	
◦ DIY HTTP	625
◦ HTTP via DownloadManager	635
◦ Using Third-Party JARs	636
◦ SSL	637
◦ Using HTTP Client Libraries	637
◦ Visit the Trails	651
• Intents, Intent Filters	
◦ What's Your Intent?	653
◦ Stating Your Intent(ions)	655
◦ Responding to Implicit Intents	655
◦ Requesting Implicit Intents	658
◦ ShareActionProvider	662
• Broadcasts and Broadcast Receivers	
◦ The Stopped State	667
◦ Example System Broadcasts	668
◦ The Order of Things	676
◦ Keeping It Local	677
◦ Visit the Trails!	678
• Tutorial #15 - Sharing Your Notes	
◦ Step #1: Adding a ShareActionProvider	679
◦ Step #2: Sharing the Note	680
◦ Step #3: Testing the Result	682
◦ In Our Next Episode...	683
• Services and the Command Pattern	
◦ Why Services?	685
◦ Setting Up a Service	686
◦ Communicating To Services	688
◦ Scenario: The Music Player	690
◦ Communicating From Services	693
◦ Scenario: The Downloader	695
• Tutorial #16 - Updating the Book	
◦ Step #1: Adding a Stub DownloadCheckService	701

◦ Step #2: Tying the Service Into the Action Bar	703
◦ Step #3: Defining Our Event	704
◦ Step #4: Defining Our JSON	705
◦ Step #5: Defining Our Retrofit Interface	706
◦ Step #6: Retrieving Our JSON Via Retrofit	707
◦ Step #7: Downloading the Update	708
◦ Step #8: Unpacking the Update	710
◦ Step #9: Using the Update	714
◦ In Our Next Episode...	719
• AlarmManager and the Scheduled Service Pattern	
◦ Scenarios	721
◦ Options	722
◦ A Simple Example	724
◦ The Five set...() Varieties	726
◦ The Four Types of Alarms	727
◦ When to Schedule Alarms	728
◦ Archetype: Scheduled Service Polling	730
◦ Staying Awake at Work	733
◦ Warning: Not All Android Devices Play Nice	737
◦ Debugging Alarms	738
◦ WakefulBroadcastReceiver	741
• Tutorial #17 - Periodic Book Updates	
◦ Step #1: Adding a Stub UpdateReceiver	745
◦ Step #2: Scheduling the Alarms	747
◦ Step #3: Adding the WakefulIntentService	748
◦ Step #4: Using WakefulIntentService	750
◦ Step #5: Completing the UpdateReceiver	750
◦ In Our Next Episode...	751
• Notifications	
◦ What's a Notification?	753
◦ Showing a Simple Notification	756
◦ The Activity-Or-Notification Scenario	761
◦ Big (and Rich) Notifications	761
◦ Foreground Services	768
◦ Disabled Notifications	771
• Tutorial #18 - Notifying the User	
◦ Step #1: Raising the Notification	775
◦ In Our Next Episode...	777
• Large-Screen Strategies and Tactics	
◦ Objective: Maximum Gain, Minimum Pain	779
◦ The Fragment Strategy	779

◦ Fragment Example: The List-and-Detail Pattern	788
◦ Other Master-Detail Strategies	800
◦ Showing More Pages	812
◦ Fragment FAQs	818
◦ Screen Size and Density Tactics	819
◦ Other Considerations	822
• Tutorial #19 - Supporting Large Screens	
◦ Step #1: Creating Our Layouts	827
◦ Step #2: Loading Our Sidebar Widgets	833
◦ Step #3: Opening the Sidebar	834
◦ Step #4: Loading Content Into the Sidebar	835
◦ Step #5: Removing Content From the Sidebar	837
• Backwards Compatibility Strategies and Tactics	
◦ Think Forwards, Not Backwards	845
◦ Aim Where You Are Going	847
◦ A Target-Rich Environment	847
◦ Lint: It's Not Just For Belly Buttons	848
◦ A Little Help From Your Friends	849
◦ Avoid the New on the Old	849
◦ Testing	853
◦ Keeping Track of Changes	853
• Getting Help	
◦ Questions. Sometimes, With Answers.	855
◦ Heading to the Source	856
◦ Getting Your News Fix	857
• Working with Library Projects	
◦ Prerequisites	859
◦ Creating a Library Project	859
◦ Using a Library Project	308
◦ What You Get	864
◦ Library Projects and the Manifest	864
◦ Limitations of Library Projects	865
• Gradle and Eclipse Projects	
◦ Prerequisites and Warnings	867
◦ "Legacy"?	867
◦ Creating Your Gradle Build File	868
◦ Examining the Gradle File	873
• Gradle and Tasks	
◦ Key Build-Related Tasks	875
◦ Results	877
• Gradle and the New Project Structure	

◦ Prerequisites and Warnings	879
◦ Objectives of the New Project Structure	880
◦ Terminology	880
◦ Creating a Project in the New Structure	884
◦ <i>What the New Project Structure Looks Like</i>	884
◦ <i>Configuring the Stock Build Types</i>	887
◦ <i>Adding Build Types</i>	892
◦ <i>Adding Product Flavors and Getting Build Variants</i>	893
◦ Doing the Splits	898
◦ <i>Revisiting the Legacy Gradle File</i>	900
◦ <i>Working with the New Project Structure in Android Studio</i> ..	902
• Gradle and Dependencies	
◦ Prerequisites and Warnings	905
◦ “Dependencies”?	906
◦ The Dependencies Closure... and the Other Dependencies Closure	906
◦ Depending Upon a JAR	906
◦ Depending Upon NDK Binaries	908
◦ <i>Depending Upon an Android Library Project</i>	908
◦ <i>Depending Upon Sub-Projects</i>	909
◦ <i>Depending Upon Artifacts</i>	911
◦ Creating Android JARs from Gradle	918
◦ A Property of Transitive (Dependencies)	919
◦ Dependencies By Build Type	919
◦ Dependencies By Flavor	920
◦ Examining Some CWAC Builds	921
• Manifest Merger Rules	
◦ <i>Prerequisites</i>	927
◦ <i>Manifest Scenarios</i>	928
◦ Pieces of Manifest Generation	929
◦ Examining the Merger Results	931
◦ <i>Merging Elements and Attributes</i>	932
◦ Employing Placeholders	938
• Signing Your App	
◦ Prerequisites	943
◦ Role of Code Signing	943
◦ What Happens In Debug Mode	944
◦ <i>Production Signing Keys</i>	945
• Distribution	
◦ Prerequisites	959
◦ <i>Get Ready To Go To Market</i>	959

• Advanced Gradle for Android Tips	
◦ <i>Prerequisites</i>	965
◦ <i>Gradle, DRY</i>	965
◦ Automating APK Version Information	971
◦ Adding to BuildConfig	974
◦ <i>Down and Dirty with the DSL</i>	975
• JUnit and Android	
◦ <i>Prerequisites</i>	979
◦ JUnit Basics	979
◦ Pondering Gradle for Android	980
◦ Where Your Test Code Lives	981
◦ Where Your Test Code Runs	985
◦ <i>Writing Your Test Cases</i>	986
◦ Your Test Suite	991
◦ Running Your Tests	992
◦ Testing Android Library Projects	998
◦ Test Dependencies	999
◦ Testing Legacy Project Structures with Gradle for Android	1000
• MonkeyRunner and the Test Monkey	
◦ Prerequisites	1003
◦ MonkeyRunner	1003
◦ Monkeying Around	1005
• Testing with UIAutomator	
◦ Prerequisites	1007
◦ What Is UIAutomator?	1007
◦ Why Choose UIAutomator Over Alternatives?	1008
◦ Creating Some Tests	1008
◦ Running Your Tests	1017
◦ Finding Your Widgets	1018
• 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
◦ <i>DatePickerDialog and TimePickerDialog</i>	1041
◦ AlertDialog	1047
◦ DialogFragments	1048
◦ DialogFragment: The Other Flavor	1052
◦ Dialogs: Modal, Not Blocking	1053
• Advanced ListView	
◦ 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	1068
• Action Bar Navigation	
◦ Prerequisites	1073
◦ List Navigation	1073
◦ Tabs (And Sometimes List) Navigation	1078
◦ Custom Navigation	1084
• Action Modes and Context Menus	
◦ Prerequisites	1086
◦ Another Wee Spot O' History	1086
◦ <i>Manual Action Modes</i>	1087
◦ <i>Multiple-Choice-Modal Action Modes</i>	1091
◦ Long-Click To Initiate an Action Mode	1095
◦ Split Action Modes	1100
• Other Advanced Action Bar Techniques	
◦ Prerequisites	1105
◦ <i>Splitting the Bar</i>	1105
◦ <i>Floating Action Bars</i>	1109
◦ Action Layouts, Action Views, and Action Providers	1111
◦ <i>Searching with SearchView</i>	1112
• AppCompat: The Official Action Bar Backport	
◦ Prerequisites	1119
◦ <i>Ummm... Why?</i>	1119
◦ <i>The Basics of Using AppCompat</i>	1121
◦ <i>Other AppCompat Effects</i>	1128
◦ <i>And Then, There Are the Bugs</i>	1134
◦ <i>To Material, or Not to Material</i>	1135
• ActionBarSherlock	
◦ Prerequisites	1139
◦ Installation	1140
◦ Code Changes	1140

• Implementing a Navigation Drawer	
◦ Prerequisites	1143
◦ What is a Navigation Drawer?	1143
◦ A Simple Navigation Drawer	1145
◦ Alternative Row Layouts	1151
◦ Additional Considerations	1153
◦ What Should Not Be in the Drawer	1162
◦ Independent Implementations	1162
• Advanced Uses of WebView	
◦ Prerequisites	1165
◦ Friends with Benefits	1165
◦ Turnabout is Fair Play	1171
◦ Navigating the Waters	1175
◦ Settings, Preferences, and Options (Oh, My!)	1176
◦ <i>Security and Your WebView</i>	1176
• The Input Method Framework	
◦ Prerequisites	1181
◦ Keyboards, Hard and Soft	1181
◦ Tailored To Your Needs	1182
◦ Tell Android Where It Can Go	1187
◦ Fitting In	1189
◦ Jane, Stop This Crazy Thing!	1191
• Fonts	
◦ Prerequisites	1193
◦ Love The One You're With	1193
◦ Yeah, But Do We Really Have To Do This in Java?	1197
◦ Here a Glyph, There a Glyph	1198
• Rich Text	
◦ Prerequisites	1201
◦ The Span Concept	1201
◦ Loading Rich Text	1203
◦ Editing Rich Text	1205
◦ Saving Rich Text	1210
◦ Manipulating Rich Text	1211
• Animators	
◦ Prerequisites	1213
◦ ViewPropertyAnimator	1213
◦ The Foundation: Value and Object Animators	1218
◦ Animating Custom Types	1221
◦ Hardware Acceleration	1222
◦ The Three-Fragment Problem	1223

• Legacy Animations	
◦ Prerequisites	1235
◦ It's Not Just For Toons Anymore	1235
◦ A Quirky Translation	1236
◦ Fading To Black. Or Some Other Color.	1240
◦ When It's All Said And Done	1242
◦ Loose Fill	1243
◦ Hit The Accelerator	1243
◦ Animate. Set. Match.	1244
◦ Active Animations	1245
• Custom Drawables	
◦ Prerequisites	1247
◦ ColorDrawable	1248
◦ AnimationDrawable	1248
◦ StateListDrawable	1251
◦ ColorStateList	1253
◦ LayerDrawable	1254
◦ TransitionDrawable	1256
◦ LevelListDrawable	1256
◦ ScaleDrawable and ClipDrawable	1258
◦ InsetDrawable	1267
◦ ShapeDrawable	1268
◦ BitmapDrawable	1278
◦ Composite Drawables	1285
◦ A Stitch In Time Saves Nine	1289
• Mapping with Maps V2	
◦ Prerequisites	1299
◦ A Brief History of Mapping on Android	1300
◦ Where You Can Use Maps V2	1301
◦ Licensing Terms for Maps V2	1301
◦ What You Need to Start	1302
◦ The Book Samples... And You!	1306
◦ Setting Up a Basic Map	1306
◦ Playing with the Map	1311
◦ Map Tiles	1314
◦ Placing Simple Markers	1314
◦ Seeing All the Markers	1317
◦ Flattening and Rotating Markers	1319
◦ Sprucing Up Your "Info Windows"	1323
◦ Images and Your Info Window	1328
◦ Setting the Marker Icon	1334

◦ <i>Responding to Taps</i>	1335
◦ <i>Dragging Markers</i>	1337
◦ The “Final” Limitations	1339
◦ A Bit More About IPC	1342
◦ <i>Finding the User</i>	1343
◦ <i>Drawing Lines and Areas</i>	1347
◦ Gestures and Controls	1350
◦ <i>Tracking Camera Changes</i>	1351
◦ <i>Maps in Fragments and Pagers</i>	1353
◦ <i>Animating Marker Movement</i>	1358
◦ Maps, of the Indoor Variety	1367
◦ Taking a Snapshot of a Map	1367
◦ MapFragment vs. MapView	1368
◦ About That AbstractMapActivity Class... ..	1369
◦ Helper Libraries for Maps V2	1373
◦ Problems with Maps V2 at Runtime	1377
◦ Problems with Maps V2 Deployment	1377
◦ What Non-Compliant Devices Show	1377
◦ Mapping Alternatives	1378
◦ News and Getting Help	1378
• Crafting Your Own Views	1381
◦ Prerequisites	1381
◦ Pick Your Poison	1381
◦ Colors, Mixed How You Like Them	1382
◦ ReverseChronometer: Simply a Custom Subclass	1393
◦ AspectLockedFrameLayout: A Custom Container	1398
◦ Mirror and MirroringFrameLayout: Draw It Yourself	1402
• Custom Dialogs and Preferences	1413
◦ Prerequisites	1413
◦ Your Dialog, Chocolate-Covered	1413
◦ Preferring Your Own Preferences, Preferably	1417
• Progress Indicators	1425
◦ Prerequisites	1425
◦ Progress Bars	1425
◦ ProgressBar and Threads	1428
◦ Tailoring Progress Bars	1431
◦ Progress Dialogs	1439
◦ Title Bar and Action Bar Progress Indicators	1441
◦ Direct Progress Indication	1443
• Advanced Notifications	1445
◦ Prerequisites	1445

◦ Being a Good Citizen	1445
◦ Wear? There!	1446
◦ Stacking Notifications	1451
◦ Avoiding Wear	1457
◦ Other Wear-Specific Notification Options	1458
◦ Lockscreen Notifications	1471
◦ Priority, and Heads-Up Notifications	1480
◦ Full-Screen Notifications	1482
◦ Custom Views: or How Those Progress Bars Work (Sometimes)	1485
◦ Seeing It In Action	1486
◦ How You Really Do Progress Notifications	1492
◦ Life After Delete	1495
◦ The Mysterious Case of the Missing Number	1496
• More Fun with Pagers	
◦ Prerequisites	1497
◦ Hosting ViewPager in a Fragment	1497
◦ Pages and the Action Bar	1499
◦ ViewPagers and Scrollable Contents	1501
◦ Using ViewPagerIndicator	1502
◦ Columns for Large, Pages for Small	1506
◦ Introducing ArrayPagerAdapter	1512
◦ Columns for Large Landscape, Pages for the Rest	1515
◦ Adding, Removing, and Moving Pages	1520
◦ <i>Inside ArrayPagerAdapter</i>	1524
• Focus Management and Accessibility	
◦ Prerequisites	1537
◦ Prepping for Testing	1538
◦ Controlling the Focus	1538
◦ Accessibility and Focus	1547
◦ Accessibility Beyond Focus	1548
◦ Accessibility Beyond Impairment	1558
• Miscellaneous UI Tricks	
◦ Prerequisites	1561
◦ Full-Screen and Lights-Out Modes	1561
◦ <i>Offering a Delayed Timeout</i>	1572
• Event Buses Alternatives	
◦ Prerequisites	1577
◦ A Brief Note About the Sample Apps	1577
◦ Standard Intents as Event Bus	1577
◦ LocalBroadcastManager as Event Bus	1578
◦ Square's Otto	1588

◦ <i>Revisiting greenrobot's EventBus</i>	1594
• Home Screen App Widgets	
◦ Prerequisites	1601
◦ East is East, and West is West... ..	1602
◦ The Big Picture for a Small App Widget	1602
◦ Crafting App Widgets	1603
◦ Another and Another	1610
◦ App Widgets: Their Life and Times	1611
◦ Controlling Your (App Widget's) Destiny	1611
◦ Change Your Look	1612
◦ One Size May Not Fit All	1613
◦ Lockscreen Widgets	1620
◦ Preview Images	1626
◦ Being a Good Host	1628
• Adapter-Based App Widgets	
◦ Prerequisites	1629
◦ AdapterViews for App Widgets	1629
◦ Building Adapter-Based App Widgets	1630
• Content Provider Theory	
◦ Prerequisites	1645
◦ <i>Using a Content Provider</i>	1645
◦ Building Content Providers	1652
◦ Issues with Content Providers	1659
• Content Provider Implementation Patterns	
◦ Prerequisites	1661
◦ The Single-Table Database-Backed Content Provider	1661
◦ The Local-File Content Provider	1669
◦ The Protected Provider	1676
◦ The Stream Provider	1679
◦ FileProvider	1682
◦ StreamProvider	1686
• The Loader Framework	
◦ Prerequisites	1689
◦ Cursors: Issues with Management	1690
◦ Introducing the Loader Framework	1690
◦ Honeycomb... Or Not	1692
◦ Using CursorLoader	1692
◦ What Else Is Missing?	1694
◦ Issues, Issues, Issues	1695
◦ Loaders Beyond Cursors	1695
◦ What Happens When...?	1695

• The ContactsContract Provider	
◦ Prerequisites	1699
◦ Introducing You to Your Contacts	1700
◦ Pick a Peck of Pickled People	1701
◦ Spin Through Your Contacts	1704
◦ Makin' Contacts	1713
• The CalendarContract Provider	
◦ Prerequisites	1720
◦ You Can't Be a Faker	1720
◦ Do You Have Room on Your Calendar?	1720
◦ Penciling In an Event	1725
• The MediaStore Provider	
◦ Prerequisites	1727
◦ What Is the MediaStore?	1728
◦ MediaStore and "Other" External Storage	1729
◦ How Does My Content Get Indexed?	1730
◦ How Do I Retrieve Video from the MediaStore?	1730
• Consuming Documents	
◦ Prerequisites	1739
◦ The Storage Access... What?	1739
◦ The Storage Access Framework Participants	1741
◦ Picking How to Pick (a Peck of Pickled Pepper Photos)	1741
◦ Opening a Document	1742
◦ The Rest of the CRUD	1745
◦ Pondering Persistent Permissions	1746
• Providing Documents	
◦ Prerequisites	1749
◦ Have Your Content, and Provide it Too	1749
◦ Key Provider Concepts	1751
◦ Pieces of a Provider	1752
◦ Optional Provider Capabilities	1766
• Encrypted Storage	
◦ Prerequisites	1774
◦ Scenarios for Encryption	1774
◦ Obtaining SQLCipher	1775
◦ Attaching SQLCipher To Your Project	1775
◦ Using SQLCipher	1776
◦ SQLCipher Limitations	1779
◦ Passwords and Sessions	1780
◦ About Those Passphrases...	1780
◦ Encrypted Preferences	1788

◦ IOCipher	1790
• Packaging and Distributing Data	
◦ Prerequisites	1791
◦ Packing a Database To Go	1791
• Advanced Database Techniques	
◦ Prerequisites	1795
◦ Full-Text Indexing	1795
• Miscellaneous Network Capabilities	
◦ Prerequisites	1811
◦ Downloading Files	1811
• Audio Playback	
◦ Prerequisites	1825
◦ Get Your Media On	1825
◦ MediaPlayer for Audio	1826
◦ Other Ways to Make Noise	1832
• Audio Recording	
◦ Prerequisites	1835
◦ Recording by Intent	1835
◦ Recording to Files	1838
◦ Recording to Streams	1841
◦ Raw Audio Input	1844
◦ Requesting the Microphone	1844
• Video Playback	
◦ Prerequisites	1847
◦ Moving Pictures	1847
• Using the Camera via 3rd-Party Apps	
◦ Prerequisites	1853
◦ Being Specific About Features	1853
◦ Still Photos: Letting the Camera App Do It	1854
◦ Scanning with ZXing	1856
◦ Videos: Letting the Camera App Do It	1857
◦ Directly Working with the Camera	1859
• Working Directly with the Camera	
◦ Prerequisites	1862
◦ Basic CameraFragment Usage	1862
◦ Simple Configuration and Usage	1863
◦ Core Camera Concepts	1870
◦ Advanced CWAC-Camera Features	1893
• Media Routes	
◦ Prerequisites	1901
◦ Terminology	1901

◦ A Tale of Three MediaRouters	1902
◦ Attaching to MediaRouter	1904
◦ User Route Selection with MediaRouteActionProvider	1905
◦ Using Live Audio Routes	1922
◦ Using Live Video Routes	1922
◦ Using Remote Playback Routes	1922
• Supporting External Displays	
◦ Prerequisites	1941
◦ A History of external displays	1941
◦ What is a Presentation?	1942
◦ Playing with External Displays	1943
◦ Detecting Displays	1949
◦ A Simple Presentation	1950
◦ A Simpler Presentation	1955
◦ Presentations and Configuration Changes	1960
◦ Presentations as Fragments	1961
◦ Another Sample Project: Slides	1971
◦ Device Support for Presentation	1978
◦ Presentations from a Service	1979
◦ Hey, What About Chromecast?	1982
• Google Cast and Chromecast	
◦ Prerequisites	1983
◦ Here a Cast, There a Cast	1983
◦ Common Chromecast Development Notes	1985
◦ Your API Choices	1985
◦ Senders and Receivers	1986
◦ Supported Media Types	1987
◦ Cast SDK Dependencies	1988
◦ Developing Google Cast Apps	1990
• The “Ten-Foot UI”	
◦ Prerequisites	1991
◦ What is the “Ten-Foot UI”?	1992
◦ Overscan	1992
◦ Navigation	1995
◦ Stylistic Considerations	1995
◦ Testing Your Theories	1997
• Creating a MediaRouteProvider	
◦ Prerequisites	1999
◦ Terminology	1999
◦ DIY Chromecast	2000
◦ Creating the MediaRouteProvider	2002

◦ Consuming the MediaRouteProvider	2012
◦ Implementing This “For Realz”	2015
• SSL	
◦ Prerequisites	2019
◦ Basic SSL Operation	2019
◦ Common SSL Problems	2020
◦ TrustManagers	2021
◦ TrustManagerBuilder	2024
◦ About That Man in the Middle	2027
◦ Self-Signed Certificates, Revisited	2029
◦ Certificate Memorizing	2030
◦ Pinning	2034
◦ NetCipher	2035
• Advanced Permissions	
◦ Prerequisites	2037
◦ Securing Yourself	2037
◦ Signature Permissions	2040
◦ The Custom Permission Vulnerability	2042
• Restricted Profiles and UserManager	
◦ Prerequisites	2053
◦ Android Tablets and Multiple User Accounts	2053
◦ Determining What the User Can Do	2059
◦ Impacts of Device-Level Restrictions	2062
◦ Enabling Custom Restrictions	2062
◦ Implicit Intents May Go “Boom”	2073
◦ The Future: App Ops?	2073
• Tapjacking	
◦ Prerequisites	2075
◦ What is Tapjacking?	2075
◦ Detecting Potential Tapjackers	2080
◦ Defending Against Tapjackers	2082
◦ Why Is This Being Discussed?	2085
◦ What Changed in 4.0.3?	2086
• Miscellaneous Security Techniques	
◦ Prerequisites	2087
◦ Public Key Validation	2087
◦ Choosing Your Signing Keysize	2103
◦ Avoiding Accidental APIs	2104
◦ Other Ways to Expose Data	2109
• Accessing Location-Based Services	
◦ Prerequisites	2113

◦ Location Providers: They Know Where You're Hiding	2114
◦ Finding Yourself	2114
◦ On the Move	2116
◦ Are We There Yet? Are We There Yet? Are We There Yet?	2117
◦ Testing... Testing...	2118
◦ Alternative Flavors of Updates	2119
◦ The Fused Option	2120
• The Fused Location Provider	
◦ Prerequisites	2121
◦ Why Use the Fused Location Provider?	2121
◦ <i>Why Not Use the Fused Location Provider?</i>	2122
◦ <i>Finding Our Location, Once</i>	2122
◦ Requesting Location Updates	2131
◦ Gaps in the Fused Location Provider	2133
• Working with the Clipboard	
◦ Prerequisites	2135
◦ Using the Clipboard on Android 1.x/2.x	2135
◦ Advanced Clipboard on Android 3.x and Higher	2138
◦ Monitoring the Clipboard	2143
◦ The Android 4.3 Clipboard Bug	2145
• Telephony	
◦ Prerequisites	2147
◦ Report To The Manager	2148
◦ You Make the Call!	2148
◦ No, Really, You Make the Call!	2151
• Working With SMS	
◦ Prerequisites	2153
◦ Sending Out an SOS, Give or Take a Letter	2154
◦ Monitoring and Receiving SMS	2161
◦ The SMS Inbox	2167
◦ Asking to Change the Default	2168
◦ SMS and the Emulator	2169
• NFC	
◦ Prerequisites	2171
◦ What Is NFC?	2171
◦ To NDEF, Or Not to NDEF	2173
◦ NDEF Modalities	2173
◦ NDEF Structure and Android's Translation	2174
◦ The Reality of NDEF	2175
◦ Sources of Tags	2177
◦ Writing to a Tag	2177

◦ Responding to a Tag	2185
◦ Expected Pattern: Bootstrap	2186
◦ Mobile Devices are Mobile	2187
◦ Enabled and Disabled	2187
◦ Android Beam	2188
◦ Beaming Files	2194
◦ Another Sample: SecretAgentMan	2196
◦ Additional Resources	2205
• Device Administration	
◦ Prerequisites	2207
◦ Objectives and Scope	2207
◦ Defining and Registering an Admin Component	2208
◦ Going Into Lockdown	2214
◦ Passwords and Device Administration	2221
◦ Getting Along with Others	2225
• PowerManager and WakeLocks	
◦ Prerequisites	2227
◦ Keeping the Screen On, UI-Style	2227
◦ The Role of the WakeLock	2228
◦ What WakefulIntentService Does	2229
• JobScheduler	
◦ Prerequisites	2231
◦ The Limitations of AlarmManager	2231
◦ Enter the JobScheduler	2232
◦ Employing JobScheduler	2232
◦ Pondering Backoff Criteria	2245
◦ Other JobScheduler Features	2246
• Push Notifications with GCM	
◦ Prerequisites	2249
◦ The Precursor: C2DM	2250
◦ The Replacement: GCM	2250
◦ The Re-Replacement: GCM 2013	2250
◦ <i>The Pieces of Push</i>	2251
◦ <i>A Simple Push</i>	2257
◦ Message Options and Advanced Features	2271
◦ Re-Registration	2273
◦ Pre-Release Features	2273
◦ Considering Encryption	2276
◦ Issues with GCM	2276
◦ Amazon Simple Notification Service and GCM	2278
• Basic Use of Sensors	

◦ Prerequisites	2279
◦ The Sensor Abstraction Model	2279
◦ Considering Rates	2280
◦ Reading Sensors	2281
◦ Batching Sensor Readings	2291
• Printing and Document Generation	
◦ Prerequisites	2294
◦ The Android Print System	2294
◦ About the Sample App	2295
◦ Printing a Bitmap	2296
◦ Printing an HTML Document	2298
◦ Printing a PDF File	2302
◦ Printing Using a Canvas	2310
◦ Print Jobs	2312
◦ Printing, Threads, and Services	2313
◦ Printing Prior to Android 4.4	2315
◦ HTML Generation	2316
◦ PDF Generation Options	2320
• Other System Settings and Services	
◦ Prerequisites	2321
◦ Setting Expectations	2321
◦ Can You Hear Me Now? OK, How About Now?	2326
◦ The Rest of the Gang	2329
• Dealing with Different Hardware	
◦ Prerequisites	2331
◦ Filtering Out Devices	2331
◦ Runtime Capability Detection	2334
◦ Dealing with Device Bugs	2335
• Responding to URLs	
◦ Prerequisites	2337
◦ Manifest Modifications	2337
◦ Creating a Custom URL	2339
◦ Reacting to the Link	2340
• Plugin Patterns	
◦ Prerequisites	2343
◦ Definitions, Scenarios, and Scope	2343
◦ The Keys to Any Plugin System	2344
◦ Case Study: DashClock	2352
◦ Other Plugin Examples	2355
• PackageManager Tricks	
◦ Prerequisites	2373

◦ Asking Around	2373
◦ Preferred Activities	2377
◦ Middle Management	2382
• Searching with SearchManager	
◦ Prerequisites	2385
◦ Hunting Season	2385
◦ Search Yourself	2387
◦ Searching for Meaning In Randomness	2394
◦ May I Make a Suggestion?	2395
◦ Putting Yourself (Almost) On Par with Google	2399
• Remote Services and the Binding Pattern	
◦ Prerequisites	2405
◦ The Binding Pattern	2406
◦ When IPC Attacks!	2412
◦ Service From Afar	2414
◦ Servicing the Service	2420
◦ Thinking About Security	2425
◦ The “Everlasting Service” Anti-Pattern	2425
• Advanced Manifest Tips	
◦ Prerequisites	2427
◦ Just Looking For Some Elbow Room	2427
◦ Using an Alias	2434
◦ Getting Meta (Data)	2436
• Miscellaneous Integration Tips	
◦ Prerequisites	2441
◦ Take the Shortcut	2441
◦ Homing Beacons for Intents	2448
• Reusable Components	
◦ Prerequisites	2449
◦ Where Do I Find Them?	2449
◦ How Are They Packaged?	2450
◦ How Do I Create Them?	2451
◦ Other Considerations for Publishing Reusable Code	2454
• The Role of Scripting Languages	
◦ Prerequisites	2459
◦ All Grown Up	2459
◦ Following the Script	2460
◦ Going Off-Script	2461
• The Scripting Layer for Android	
◦ Prerequisites	2465
◦ The Role of SL4A	2465

◦ Getting Started with SL4A	2466
◦ Writing SL4A Scripts	2474
◦ Running SL4A Scripts	2479
◦ Potential Issues	2482
• JVM Scripting Languages	
◦ Prerequisites	2485
◦ Languages on Languages	2485
◦ A Brief History of JVM Scripting	2486
◦ Limitations	2487
◦ SL4A and JVM Languages	2488
◦ Embedding JVM Languages	2488
◦ Other JVM Scripting Languages	2502
• Advanced Emulator Capabilities	
◦ Prerequisites	2505
◦ x86 Images	2505
◦ Hardware Graphics Acceleration	2508
◦ Defining New Devices	2511
◦ Keyboard Behavior	2514
◦ Headless Operation	2514
• Using Lint	
◦ Prerequisites	2515
◦ What It Is	2515
◦ When It Runs	2516
◦ What to Fix	2519
◦ What to Configure	2519
• Using Hierarchy View	
◦ Prerequisites	2525
◦ Launching Hierarchy View	2525
◦ Viewing the View Hierarchy	2526
◦ ViewServer	2529
• Using DDMS	
◦ Prerequisites	2531
◦ Starting DDMS	2531
◦ File Push and Pull	2532
◦ Screenshots	2533
◦ Location Updates	2533
◦ Placing Calls and Messages	2534
• Issues with Speed	
◦ Prerequisites	2537
◦ Getting Things Done	2537
◦ Your UI Seems... Janky	2538

◦ Not Far Enough in the Background	2538
◦ Playing with Speed	2539
• Finding CPU Bottlenecks	
◦ Prerequisites	2541
◦ Traceview	2542
◦ Other General CPU Measurement Techniques	2551
◦ UI “Jank” Measurement	2553
• Focus On: NDK	
◦ Prerequisites	2569
◦ The Role of the NDK	2570
◦ NDK Installation and Project Setup	2573
◦ Writing Your Makefile(s)	2577
◦ Building Your Library	2578
◦ Using Your Library Via JNI	2579
◦ Building and Deploying Your Project	2584
◦ Gradle and the NDK	2586
• Improving CPU Performance in Java	
◦ Prerequisites	2593
◦ Reduce CPU Utilization	2593
◦ Reduce Time on the Main Application Thread	2598
◦ Improve Throughput and Responsiveness	2606
• Finding and Eliminating Jank	
◦ Prerequisites	2609
◦ The Case: ThreePaneDemoBC	2609
◦ Are We Janky?	2610
◦ Finding the Source of the Jank	2610
◦ Where Things Went Wrong	2620
◦ Removing the Jank	2621
• Issues with Bandwidth	
◦ Prerequisites	2623
◦ You’re Using Too Much of the Slow Stuff	2624
◦ You’re Using Too Much of the Expensive Stuff	2624
◦ You’re Using Too Much of Somebody Else’s Stuff	2625
◦ You’re Using Too Much... And There Is None	2626
• Focus On: TrafficStats	
◦ Prerequisites	2627
◦ TrafficStats Basics	2627
◦ Example: TrafficMonitor	2629
◦ Other Ways to Employ TrafficStats	2637
• Measuring Bandwidth Consumption	
◦ Prerequisites	2639

◦ On-Device Measurement	2639
◦ Off-Device Measurement	2641
◦ Tactical Measurement in DDMS	2643
• Being Smarter About Bandwidth	
◦ Prerequisites	2647
◦ Bandwidth Savings	2647
◦ Bandwidth Shaping	2653
◦ Avoiding Metered Connections	2656
• Issues with Application Heap	
◦ Prerequisites	2659
◦ You Are in a Heap of Trouble	2660
◦ Determining Your Heap Size At Runtime	2660
◦ Fragments of Memory	2661
◦ Getting a Trim	2662
◦ Warning: Contains Graphic Images	2663
◦ Releasing SQLite Memory	2674
◦ In Too Deep (on the Stack)	2674
• Finding Memory Leaks with MAT	
◦ Prerequisites	2677
◦ Setting Up MAT	2677
◦ Getting Heap Dumps	2678
◦ Basic MAT Operation	2684
◦ Some Leaks and Their MAT Analysis	2691
• Issues with System RAM	
◦ Prerequisites	2699
◦ Can't We All Just Get Along?	2699
◦ Contributors to System RAM Consumption	2700
◦ Measuring System RAM Consumption: Tools	2701
◦ Measuring System RAM Consumption: Runtime	2716
◦ Learn To Let Go (Of Your Heap)	2717
• Issues with Battery Life	
◦ Prerequisites	2719
◦ You're Getting Blamed	2720
◦ Not All Batteries Are Created Equal	2721
◦ Stretching Out the Last mWh	2721
• Power Measurement Options	
◦ Prerequisites	2723
◦ batterystats and the Battery Historian	2724
◦ The Qualcomm Tool (That Must Not Be Named)	2734
◦ PowerTutor	2735
◦ Battery Screen in Settings Application	2739

◦ BatteryInfo Dump	2741
• Sources of Power Drain	
◦ Prerequisites	2745
◦ Screen	2746
◦ Disk I/O	2747
◦ WiFi and Mobile Data	2748
◦ GPS	2751
◦ Camera	2752
◦ Additional Sources	2752
• Addressing Application Size Issues	
◦ Prerequisites	2755
◦ <i>Java Code, and the 64K Method Limit</i>	2755
◦ Native Code	2760
◦ Images	2762
◦ APK Expansion Files	2763
• The Role of Alternative Environments	
◦ Prerequisites	2765
◦ In the Beginning, There Was Java... ..	2766
◦ ... And It Was OK	2766
◦ Bucking the Trend	2767
◦ Support, Structure	2767
◦ Caveat Developer	2768
• HTML5	
◦ Prerequisites	2769
◦ Offline Applications	2769
◦ Web Storage	2776
◦ Going To Production	2779
◦ Issues You May Encounter	2780
◦ HTML5: The Baseline	2783
• PhoneGap	
◦ Prerequisites	2785
◦ What Is PhoneGap?	2785
◦ Using PhoneGap	2788
◦ PhoneGap and the Checklist Sample	2793
◦ Issues You May Encounter	2798
◦ For More Information	2801
• Other Alternative Environments	
◦ Prerequisites	2803
◦ Rhodes	2803
◦ Flash, Flex, and AIR	2804
◦ JRuby and Ruboto	2804

◦ App Inventor	2805
◦ Titanium Mobile	2807
◦ Other JVM Compiled Languages	2808
• Anti-Patterns	
◦ Prerequisites	2811
◦ Leak Threads... Or Things Attached to Threads	2811
◦ Use Large Heap Unnecessarily	2813
◦ Misuse the MENU Button	2815
◦ Interfere with Navigation	2816
◦ Use android:sharedUserId	2818
◦ Implement a “Quit” Button	2819
◦ Terminate Your Process	2821
◦ Try to Hide from the User	2822
◦ Use Multiple Processes	2823
◦ Hog System Resources	2825
• Widget Catalog: AdapterViewFlipper	
◦ Key Usage Tips	2827
◦ A Sample Usage	2828
◦ Visual Representation	2828
• Widget Catalog: CalendarView	
◦ Key Usage Tips	2829
◦ A Sample Usage	2830
◦ Visual Representation	2831
• Widget Catalog: DatePicker	
◦ Key Usage Tips	2833
◦ A Sample Usage	2834
◦ Visual Representation	2835
• Widget Catalog: ExpandableListView	
◦ Key Usage Tips	2839
◦ A Sample Usage	2840
◦ Visual Representation	2846
• Widget Catalog: SeekBar	
◦ Key Usage Tips	2849
◦ A Sample Usage	2849
◦ Visual Representation	2851
• Widget Catalog: SlidingDrawer	
◦ Key Usage Tips	2853
◦ A Sample Usage	2854
◦ Visual Representation	2855
• Widget Catalog: StackView	
◦ Key Usage Tips	2857

◦ A Sample Usage	2858
◦ Visual Representation	2859
• Widget Catalog: TabHost and TabWidget	
◦ Deprecation Notes	2861
◦ Key Usage Tips	2861
◦ A Sample Usage	2862
◦ Visual Representation	2864
• Widget Catalog: TimePicker	
◦ Key Usage Tips	2867
◦ A Sample Usage	2867
◦ Visual Representation	2869
• Widget Catalog: ViewFlipper	
◦ Key Usage Tips	2871
◦ A Sample Usage	2872
◦ Visual Representation	2873
• Device Catalog: Kindle Fire	
◦ Prerequisites	2875
◦ Introducing the Kindle Fire series	2875
◦ What Features and Configurations Does It Use?	2876
◦ What Is Really Different?	2878
◦ Getting Your Development Environment Established	2884
◦ How Does Distribution Work?	2886
◦ Amazon Equivalents of Google Services	2887
◦ Getting Help with the Kindle Fire	2888
• Device Catalog: BlackBerry	
◦ I Thought BlackBerry Had Their Own OS?	2889
◦ What Else Is Different?	2890
◦ What Are We Making?	2893
◦ Getting Your Development Environment Established	2893
◦ How Does Distribution Work?	2895
• Device Catalog: Wrist Wearables	
◦ Prerequisites	2898
◦ Divvying Up the Wearables Space	2898
◦ Example Wrist Wearables	2899
◦ Strategic Considerations	2902
◦ Tactical Considerations	2904
◦ What About Android Wear?	2907
• Device Catalog: Google TV	
◦ Prerequisites	2909
◦ What Features and Configurations Does It Use?	2910
◦ What Is Really Different?	2911

◦ Getting Your Development Environment Established	2914
◦ How Does Distribution Work?	2917
◦ Getting Help	2919
• Device Catalog: Amazon Fire TV	
◦ Prerequisites	2921
◦ Introducing the Fire TV	2921
◦ What Features and Configurations Does It Use?	2926
◦ What Is Really Different?	2928
◦ Casting and Fire TV	2928
◦ Getting Your Development Environment Established	2929
◦ Working with the Remote and Controller	2931
◦ How Does Distribution Work?	2933
◦ Getting Help	2933
• CWAC Libraries	
◦ cwac-adapter	2935
◦ cwac-camera	2935
◦ cwac-colormixer	2936
◦ cwac-layouts	2936
◦ cwac-mediarouter	2936
◦ cwac-merge	2936
◦ cwac-pager	2937
◦ cwac-presentation	2937
◦ cwac-provider	2937
◦ cwac-richedit	2937
◦ cwac-sacklist	2938
◦ cwac-security	2938
◦ cwac-strictmodeex	2938
◦ cwac-wakeful	2938

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,

PREFACE

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](#)
- [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](#)
- [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](#)

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”](#)
- [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

- [Responding to URLs](#)
- [Plugin Patterns](#)
- [PackageManager Tricks](#)
- [Searching with SearchManager](#)
- [Remote Services and the Binding Pattern](#)
- [Advanced Manifest Tips](#)
- [Miscellaneous Integration Tips](#)
- [Reusable Components](#)

Scripting Languages

- [The Role of Scripting Languages](#)
- [The Scripting Layer for Android](#)
- [JVM Scripting Languages](#)

Other Tools

- [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](#)

Alternatives for App Development

- [The Role of Alternative Environments](#)
- [HTML5](#)
- [PhoneGap](#)
- [Other Alternative Environments](#)

Miscellaneous Topics

- [Anti-Patterns](#)

Widget Catalog

- [AdapterViewFlipper](#)
- [CalendarView](#)
- [DatePicker](#)
- [ExpandableListView](#)
- [SeekBar](#)

- [SlidingDrawer](#)
- [StackView](#)
- [TabHost](#)
- [TimePicker](#)
- [ViewFlipper](#)

Device Catalog

- [Kindle Fire](#)
- [BlackBerry](#)
- [Wrist Wearables](#)
- [Google TV](#)
- [Amazon Fire TV](#)

Appendices

- [Appendix A: CWAC Libraries](#)

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.

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.

PREFACE

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 December 2018*. Of course, watch the CommonsWare Web site, as this edition might be relicensed sooner based on sales.

For more details on the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license, visit [the Creative Commons Web site](#)

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

Acknowledgments

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

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

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

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.

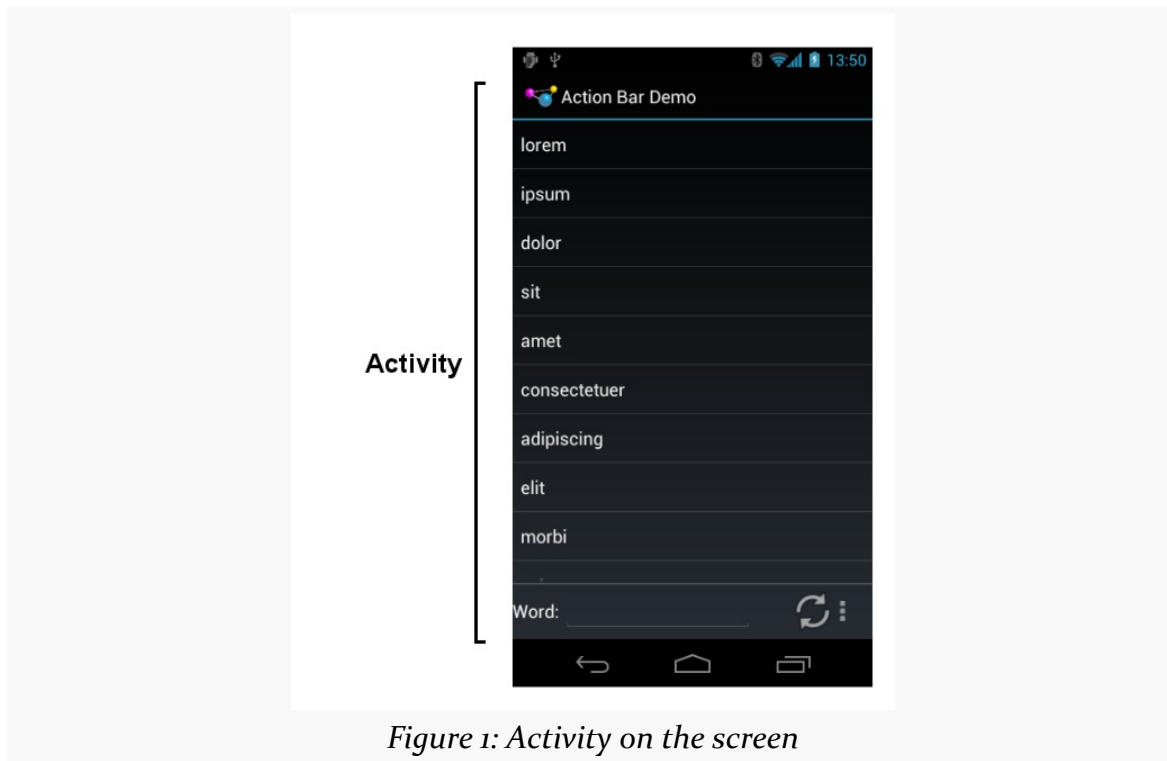
There are four types of components, all of which will be covered extensively in this book:

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



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:

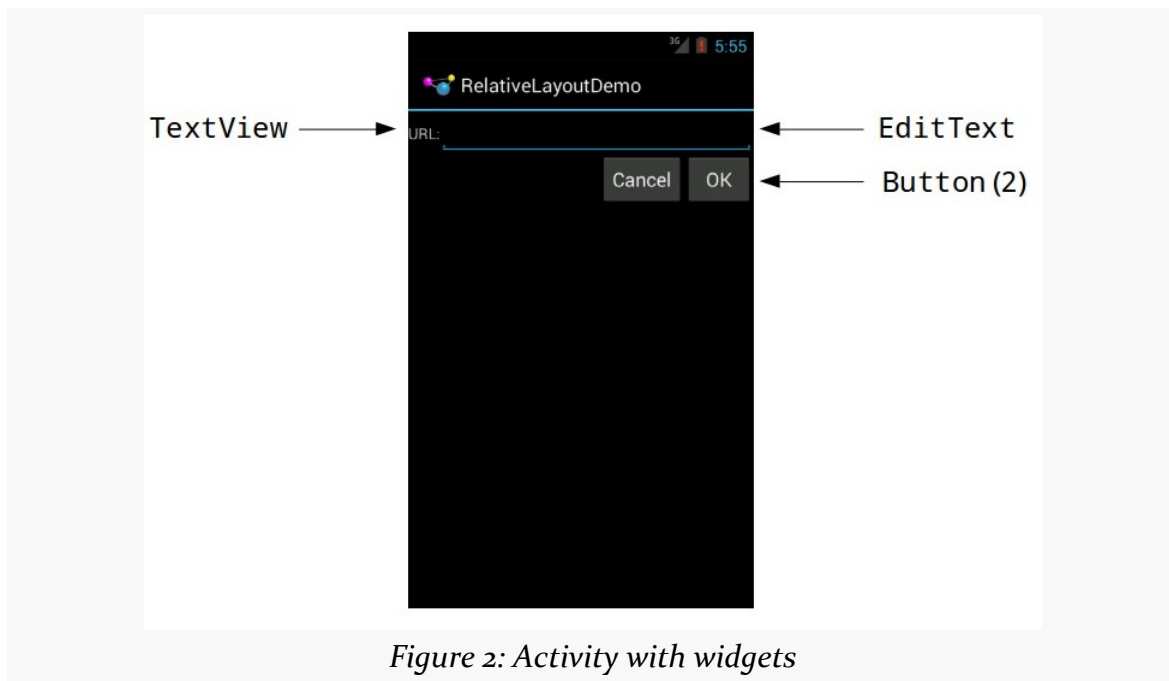


Figure 2: Activity with widgets

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

KEY ANDROID CONCEPTS

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).

Sometimes, supporting larger screens means you will want to “snap together” parts of your smaller-screen UI. For example, Gmail on a tablet will show your list of labels, the list of conversations in a selected label, and the list of messages in a selected conversation, all in one activity. However, Gmail on a phone cannot do that, as there is not enough screen space, so it shows each of those (labels, conversations, messages) in separate activities. Android supplies a construct called the *fragment* to help make it easier for you to implement these sorts of effects.

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. 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 package

3. No two applications can be uploaded to the Play Store having the same package

Note that sometimes you will see reference to an “application ID” — this is the role of the package name from the second and third items in the list.

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

KEY ANDROID CONCEPTS

future versions of Android (forwards compatibility), albeit perhaps missing some features or working in some sort of “compatibility mode”. And there are well-trodden paths for how to create apps that will work both on the latest and on previous versions of Android (backwards compatibility).

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

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

- API Level 10 (Android 2.3.3)
- API Level 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)

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 API level for most form factors is 21, representing Android 5.0. There is an API Level 20, which is the version of Android 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.

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 off of 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 and Linux, and presumably from Apple for OS X. 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 three major options that this book covers:

1. Use [Android Studio](#)
2. Use Eclipse, as a fresh install via the [ADT Bundle](#)
3. Use Eclipse, [adding in the ADT plugin to an existing Eclipse installation](#)

A fourth 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

TUTORIAL #1 - INSTALLING THE TOOLS

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 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.

Options #2 and #3 are close to being mutually exclusive. And, 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 one or more 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.

Option #2 - Android ADT Bundle

The Android ADT Bundle contains:

- The Android SDK, which gives you all the tools you need to create and test Android applications. It comes in two parts: the base tools, plus version-specific SDKs and related add-ons.
- A copy of Eclipse, with the Android Developer Tools (ADT) plug-in pre-installed.

From [the Android Developer site's SDK area](#), choose the ADT Bundle ZIP file that is appropriate for your development machine and unZIP it in some likely location.

Option #3 - 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.

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](#). 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:

TUTORIAL #1 - INSTALLING THE TOOLS

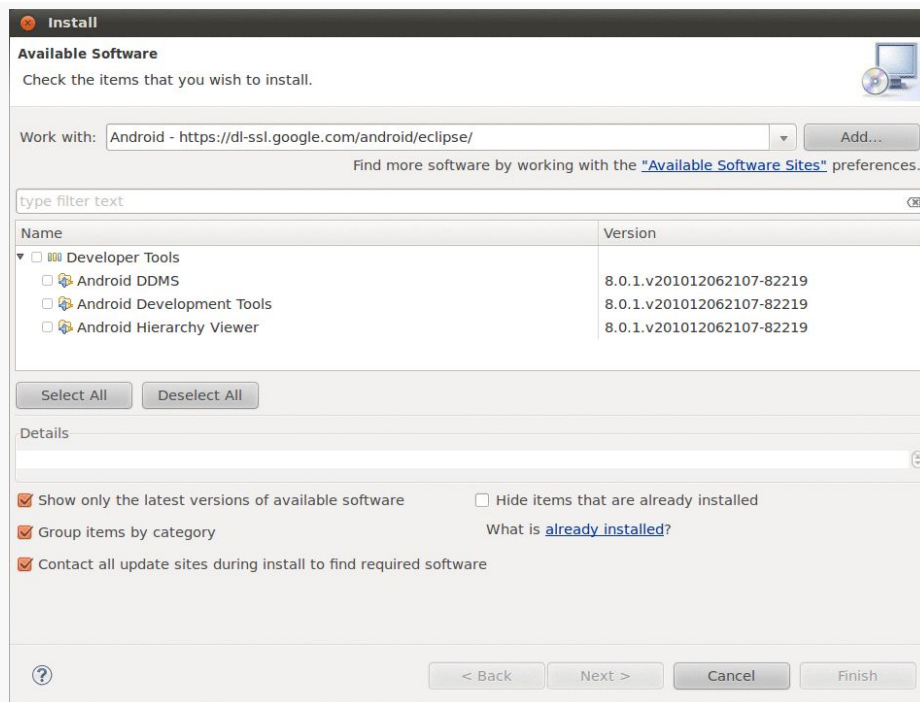


Figure 3: Eclipse ADT plug-in installation

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:

TUTORIAL #1 - INSTALLING THE TOOLS

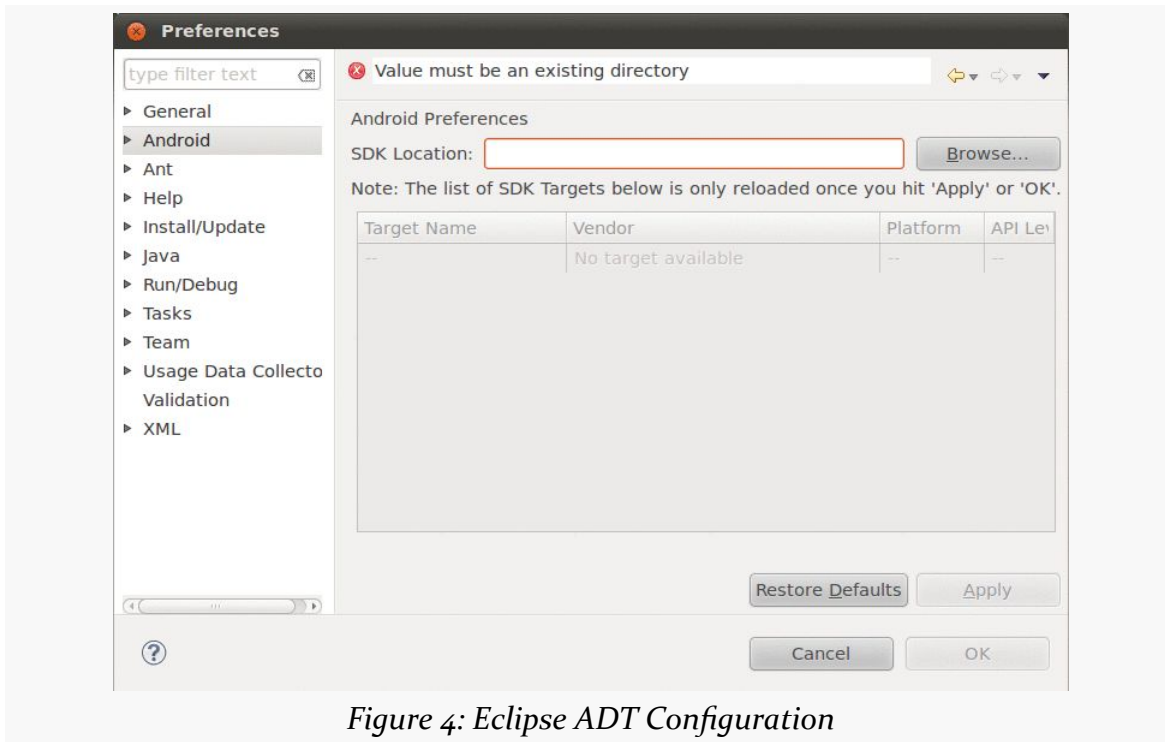


Figure 4: Eclipse ADT Configuration

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:

TUTORIAL #1 - INSTALLING THE TOOLS

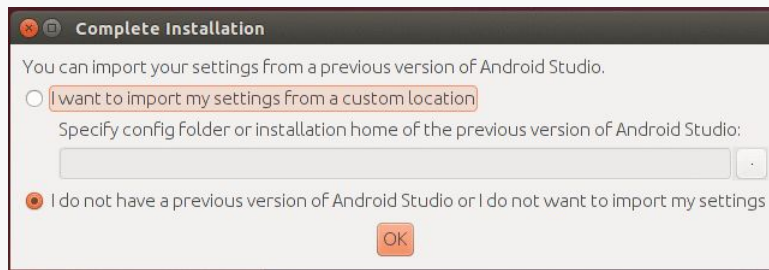


Figure 5: Android Studio First-Run Settings Migration Dialog

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:

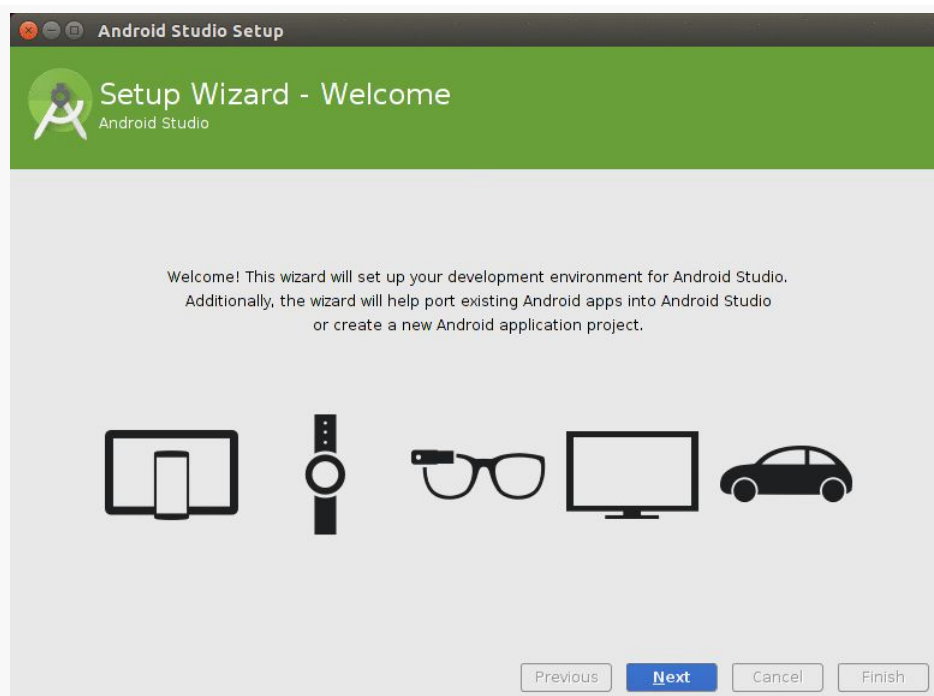


Figure 6: Android Studio Setup Wizard, First Page

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

TUTORIAL #1 - INSTALLING THE TOOLS

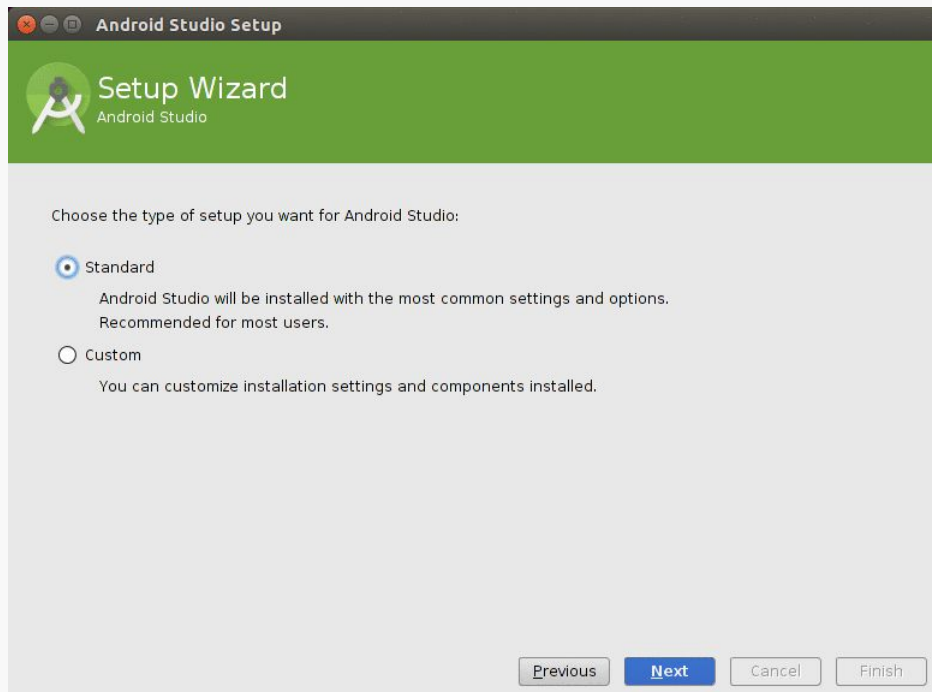


Figure 7: Android Studio Setup Wizard, Second Page

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

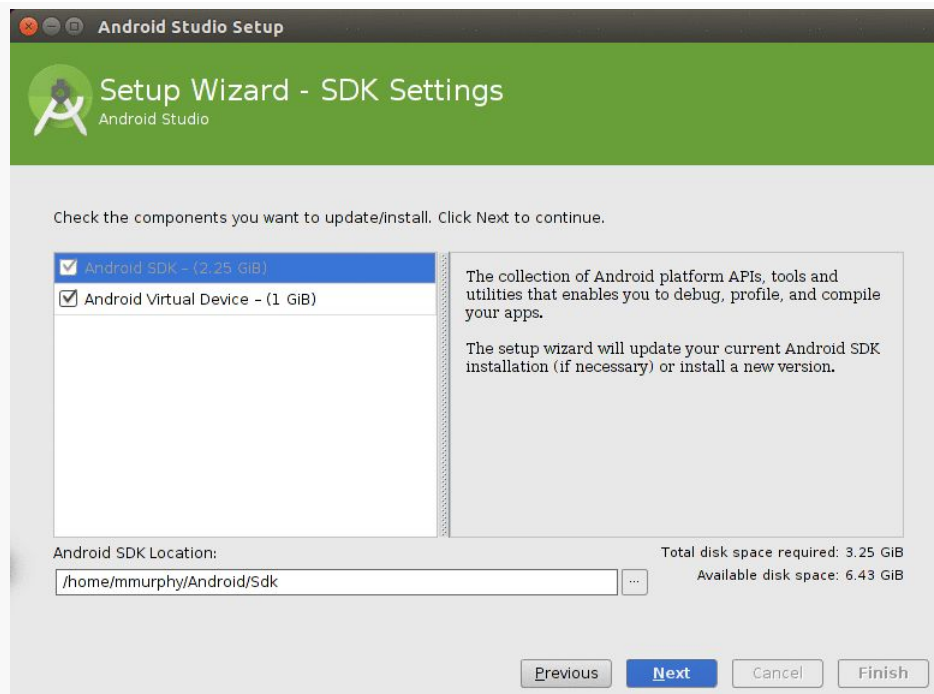


Figure 8: Android Studio Setup Wizard, Custom Page

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

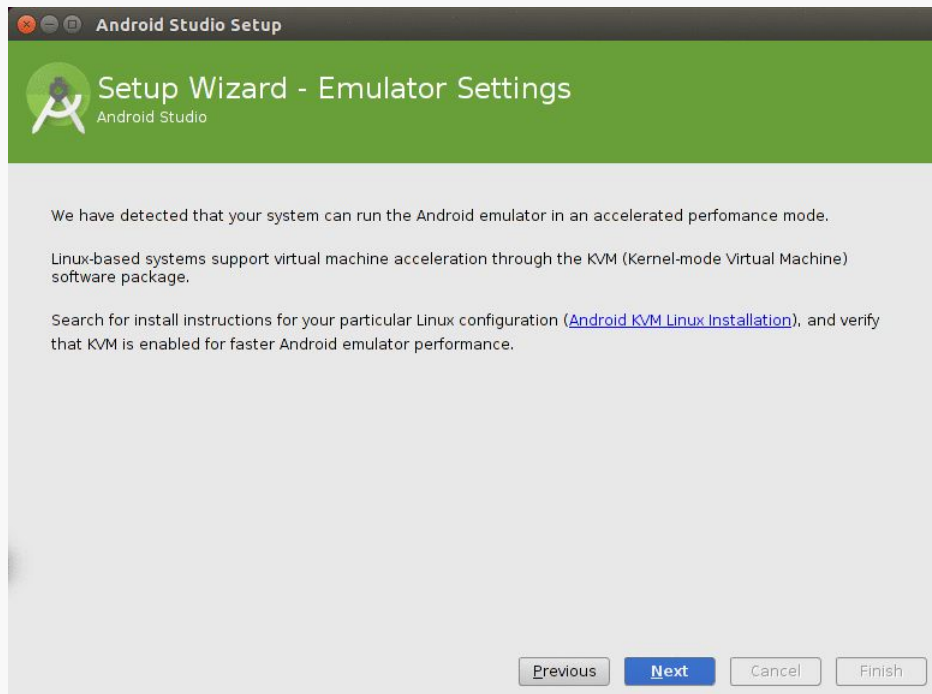


Figure 9: Android Studio Setup Wizard, Emulator Info Page

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:

TUTORIAL #1 - INSTALLING THE TOOLS

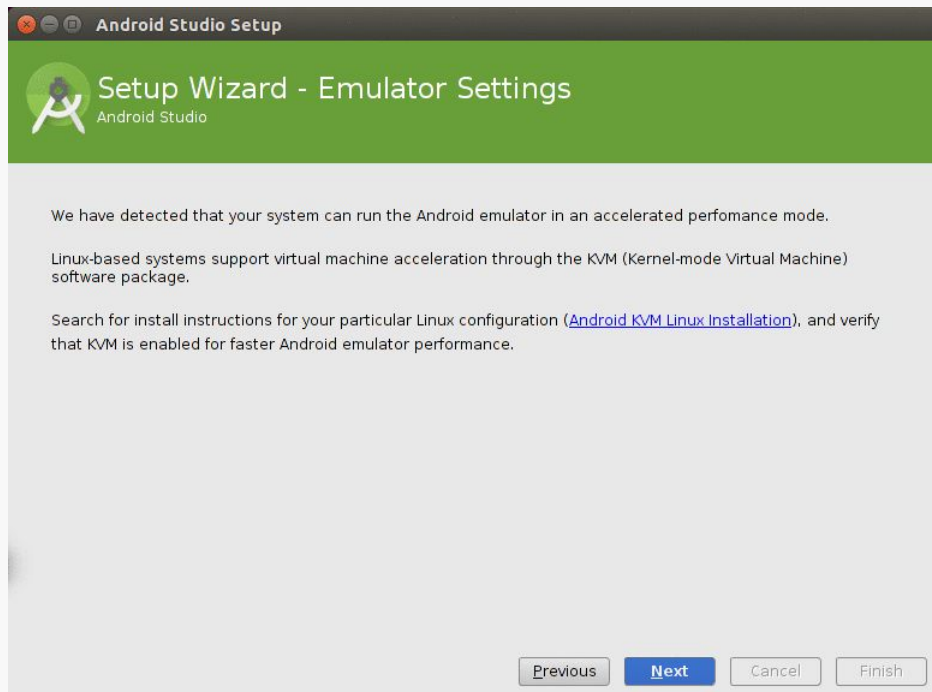


Figure 10: Android Studio Setup Wizard, License Page

Here, 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.

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

TUTORIAL #1 - INSTALLING THE TOOLS

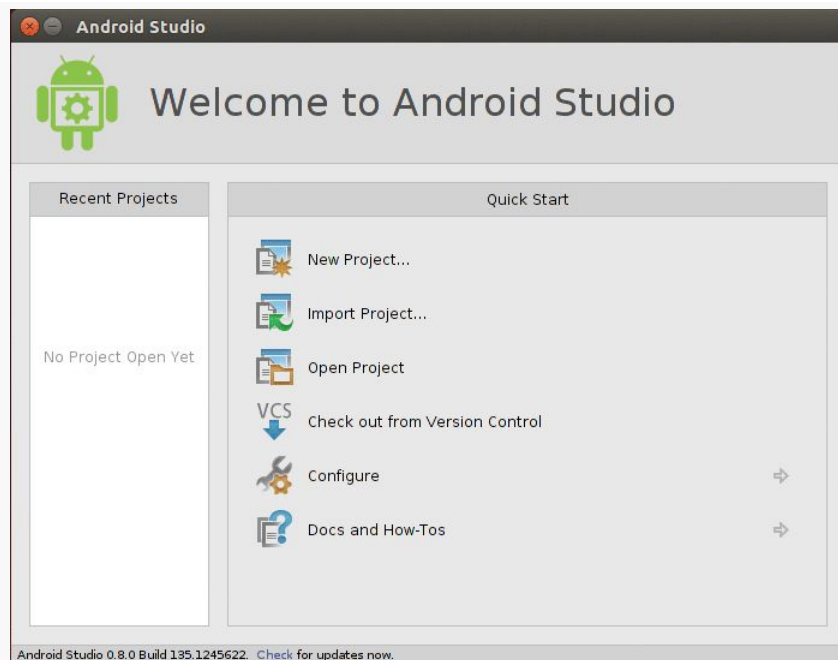


Figure 11: Android Studio Welcome Dialog

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

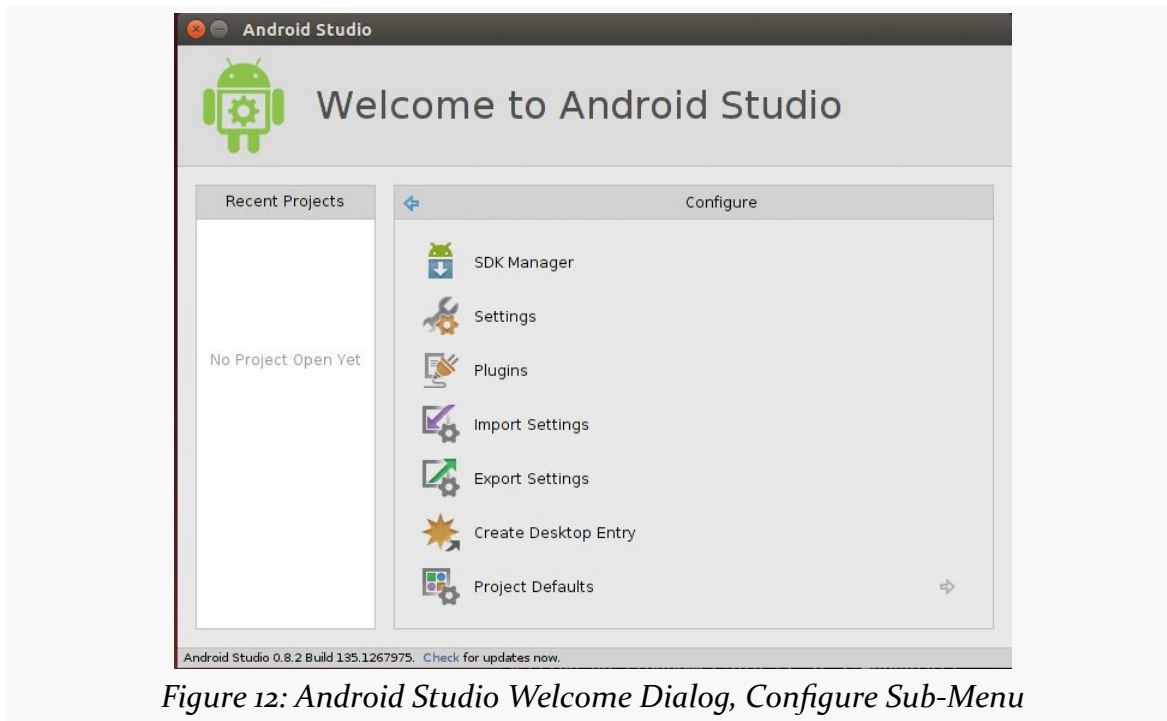


Figure 12: Android Studio Welcome Dialog, Configure Sub-Menu

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

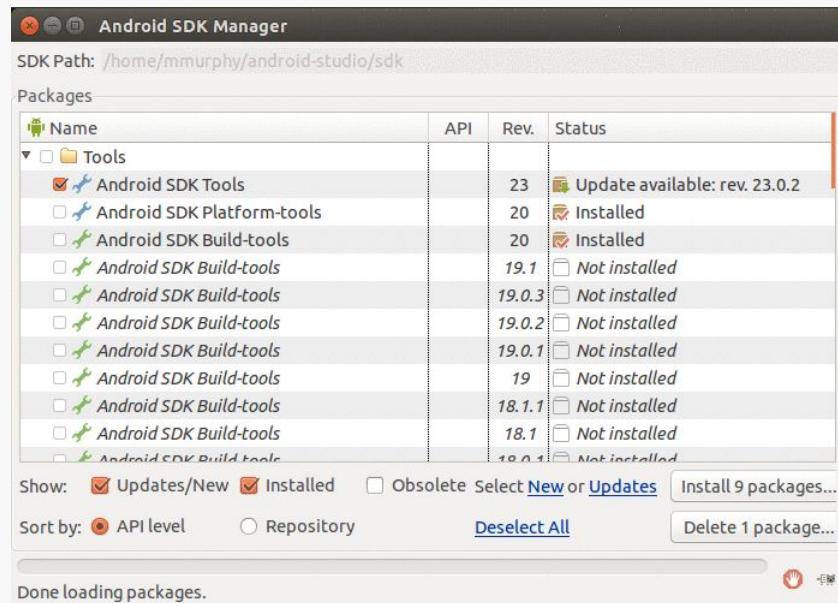


Figure 13: Android SDK Manager

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 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 the API level you chose. 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.

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

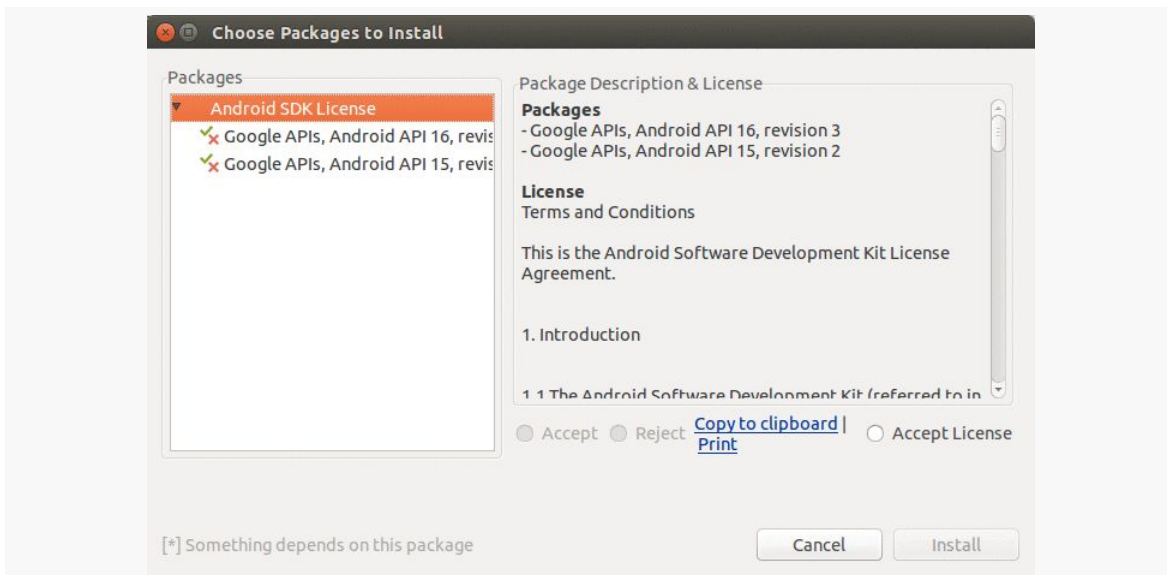


Figure 14: Android SDK Manager Installing Packages

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.

