

# Android Studio 2 Development



## Essentials

# **Android Studio 2 Development Essentials**

---

Android Studio 2 Development Essentials

ISBN-13: 978-1532853319

© 2016 Neil Smyth. All Rights Reserved.

This book is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

The content of this book is provided for informational purposes only. Neither the publisher nor the author offers any warranties or representation, express or implied, with regard to the accuracy of information contained in this book, nor do they accept any liability for any loss or damage arising from any errors or omissions.

This book contains trademarked terms that are used solely for editorial purposes and to the benefit of the respective trademark owner. The terms used within this book are not intended as infringement of any trademarks.

Rev: 1.0





# Table of Contents

<b>1. Introduction.....</b>	<b>1</b>
1.1 Downloading the Code Samples.....	2
1.2 Download the eBook.....	2
1.3 Feedback.....	2
1.4 Errata.....	2
<b>2. Setting up an Android Studio Development Environment.....</b>	<b>3</b>
2.1 System Requirements.....	3
2.2 Installing the Java Development Kit (JDK).....	3
2.2.1 <i>Windows JDK Installation</i> .....	4
2.2.2 <i>Mac OS X JDK Installation</i> .....	4
2.3 Linux JDK Installation.....	5
2.4 Downloading the Android Studio Package.....	6
2.5 Installing Android Studio.....	7
2.5.1 <i>Installation on Windows</i> .....	7
2.5.2 <i>Installation on Mac OS X</i> .....	7
2.5.3 <i>Installation on Linux</i> .....	8
2.6 The Android Studio Setup Wizard.....	9
2.7 Installing Additional Android SDK Packages.....	10
2.8 Making the Android SDK Tools Command-line Accessible.....	13
2.8.1 <i>Windows 7</i> .....	14
2.8.2 <i>Windows 8.1</i> .....	15
2.8.3 <i>Windows 10</i> .....	16
2.8.4 <i>Linux</i> .....	16
2.8.5 <i>Mac OS X</i> .....	16
2.9 Updating the Android Studio and the SDK.....	16
2.10 Summary.....	17
<b>3. Creating an Example Android App in Android Studio.....</b>	<b>19</b>
3.1 Creating a New Android Project.....	19
3.2 Defining the Project and SDK Settings.....	20
3.3 Creating an Activity.....	21
3.4 Modifying the Example Application.....	23
3.5 Reviewing the Layout and Resource Files.....	28
3.6 Previewing the Layout.....	31
3.7 Summary.....	32
<b>4. A Tour of the Android Studio User Interface.....</b>	<b>33</b>

4.1 The Welcome Screen .....	33
4.2 The Main Window .....	34
4.3 The Tool Windows .....	35
4.4 Android Studio Keyboard Shortcuts .....	39
4.5 Switcher and Recent Files Navigation.....	39
4.6 Changing the Android Studio Theme.....	41
4.7 Summary.....	41
<b>5. Creating an Android Virtual Device (AVD) in Android Studio .....</b>	<b>43</b>
5.1 About Android Virtual Devices .....	43
5.2 Creating a New AVD .....	44
5.3 Starting the Emulator .....	45
5.4 Running the Application in the AVD .....	46
5.5 Run/Debug Configurations .....	48
5.6 Stopping a Running Application.....	49
5.7 AVD Command-line Creation.....	51
5.8 Android Virtual Device Configuration Files.....	53
5.9 Moving and Renaming an Android Virtual Device .....	53
5.10 Summary.....	54
<b>6. Using and Configuring the Android Studio 2 AVD Emulator .....</b>	<b>55</b>
6.1 The Emulator Environment.....	55
6.2 The Emulator Toolbar Options .....	56
6.3 Working in Zoom Mode .....	58
6.4 Resizing the Emulator Window.....	58
6.5 Extended Control Options .....	58
6.5.1 Location.....	59
6.5.2 Cellular .....	59
6.5.3 Battery.....	59
6.5.4 Phone.....	60
6.5.5 Directional Pad .....	60
6.5.6 Fingerprint.....	60
6.5.7 Settings.....	60
6.5.8 Help .....	60
6.6 Drag and Drop Support.....	60
6.7 Configuring Fingerprint Emulation .....	61
6.8 Multi-Core Support.....	63
6.9 Summary.....	63
<b>7. Testing Android Studio Apps on a Physical Android Device .....</b>	<b>65</b>
7.1 An Overview of the Android Debug Bridge (ADB) .....	65

7.2 Enabling ADB on Android 6.0 based Devices.....	66
7.2.1 Mac OS X ADB Configuration .....	67
7.2.2 Windows ADB Configuration.....	67
7.2.3 Linux adb Configuration .....	69
7.3 Testing the adb Connection .....	70
7.4 Summary .....	71
<b>8. The Basics of the Android Studio Code Editor .....</b>	<b>73</b>
8.1 The Android Studio Editor .....	73
8.2 Splitting the Editor Window .....	76
8.3 Code Completion .....	77
8.4 Statement Completion .....	78
8.5 Parameter Information .....	79
8.6 Code Generation .....	79
8.7 Code Folding.....	80
8.8 Quick Documentation Lookup.....	82
8.9 Code Reformatting .....	83
8.10 Summary .....	83
<b>9. An Overview of the Android Architecture .....</b>	<b>85</b>
9.1 The Android Software Stack .....	85
9.2 The Linux Kernel .....	86
9.3 Android Runtime – ART .....	87
9.4 Android Libraries .....	87
9.4.1 C/C++ Libraries .....	88
9.5 Application Framework .....	88
9.6 Applications.....	89
9.7 Summary .....	89
<b>10. The Anatomy of an Android Application .....</b>	<b>91</b>
10.1 Android Activities .....	91
10.2 Android Intents.....	92
10.3 Broadcast Intents .....	92
10.4 Broadcast Receivers .....	92
10.5 Android Services.....	93
10.6 Content Providers.....	93
10.7 The Application Manifest .....	94
10.8 Application Resources .....	94
10.9 Application Context .....	94
10.10 Summary .....	94
<b>11. Understanding Android Application and Activity Lifecycles .....</b>	<b>95</b>

11.1 Android Applications and Resource Management .....	95
11.2 Android Process States .....	96
11.2.1 Foreground Process .....	96
11.2.2 Visible Process .....	96
11.2.3 Service Process .....	97
11.2.4 Background Process .....	97
11.2.5 Empty Process .....	97
11.3 Inter-Process Dependencies .....	97
11.4 The Activity Lifecycle .....	97
11.5 The Activity Stack.....	97
11.6 Activity States .....	99
11.7 Configuration Changes .....	99
11.8 Handling State Change.....	99
11.9 Summary.....	100
<b>12. Handling Android Activity State Changes.....</b>	<b>101</b>
12.1 The Activity Class .....	101
12.2 Dynamic State vs. Persistent State .....	104
12.3 The Android Activity Lifecycle Methods .....	105
12.4 Activity Lifetimes .....	106
12.5 Summary.....	107
<b>13. Android Activity State Changes by Example.....</b>	<b>109</b>
13.1 Creating the State Change Example Project .....	109
13.2 Designing the User Interface .....	110
13.3 Overriding the Activity Lifecycle Methods.....	112
13.4 Filtering the LogCat Panel.....	116
13.5 Running the Application .....	117
13.6 Experimenting with the Activity .....	118
13.7 Summary.....	119
<b>14. Saving and Restoring the State of an Android Activity .....</b>	<b>121</b>
14.1 Saving Dynamic State .....	121
14.2 Default Saving of User Interface State.....	121
14.3 The Bundle Class.....	123
14.4 Saving the State .....	123
14.5 Restoring the State .....	125
14.6 Testing the Application .....	126
14.7 Summary.....	126
<b>15. Understanding Android Views, View Groups and Layouts .....</b>	<b>127</b>
15.1 Designing for Different Android Devices .....	127



15.2 Views and View Groups .....	127
15.3 Android Layout Managers .....	128
15.4 The View Hierarchy .....	129
15.5 Creating User Interfaces .....	131
15.6 Summary .....	131
<b>16. A Guide to the Android Studio Designer Tool .....</b>	<b>133</b>
16.1 Blank vs. Empty Activity Templates .....	133
16.2 The Android Studio Designer .....	136
16.3 Design Mode .....	136
16.4 Text Mode .....	137
16.5 Setting Properties .....	138
16.6 Type Morphing .....	140
16.7 Creating a Custom Device Definition .....	140
16.8 Summary .....	141
<b>17. Designing a User Interface using the Android Studio Designer Tool .....</b>	<b>143</b>
17.1 An Android Studio Designer Tool Example .....	143
17.2 Creating a New Activity .....	143
17.3 Designing the User Interface .....	145
17.4 Editing View Properties .....	146
17.5 Running the Application .....	147
17.6 Manually Creating an XML Layout .....	147
17.7 Using the Hierarchy Viewer .....	149
17.8 Summary .....	153
<b>18. Creating an Android User Interface in Java Code .....</b>	<b>155</b>
18.1 Java Code vs. XML Layout Files .....	155
18.2 Creating Views .....	156
18.3 Properties and Layout Parameters .....	156
18.4 Creating the Example Project in Android Studio .....	157
18.5 Adding Views to an Activity .....	157
18.6 Setting View Properties .....	159
18.7 Adding Layout Parameters and Rules .....	160
18.8 Using View IDs .....	162
18.9 Converting Density Independent Pixels (dp) to Pixels (px) .....	164
18.10 Summary .....	166
<b>19. Using the Android GridLayout Manager in Android Studio Designer .....</b>	<b>169</b>
19.1 Introducing the Android GridLayout and Space Classes .....	169
19.2 The GridLayout Example .....	170
19.3 Creating the GridLayout Project .....	170

19.4 Creating the GridLayout Instance .....	170
19.5 Adding Views to GridLayout Cells .....	172
19.6 Moving and Deleting Rows and Columns .....	173
19.7 Implementing Cell Row and Column Spanning.....	174
19.8 Changing the Gravity of a GridLayout Child.....	174
19.9 Summary.....	177
<b>20. Working with the Android GridLayout using XML Layout Resources .....</b>	<b>179</b>
20.1 GridLayouts in XML Resource Files .....	179
20.2 Adding Child Views to the GridLayout .....	180
20.3 Declaring Cell Spanning, Gravity and Margins .....	182
20.4 Summary.....	184
<b>21. An Overview and Example of Android Event Handling.....</b>	<b>185</b>
21.1 Understanding Android Events.....	185
21.2 Using the android:onClick Resource .....	186
21.3 Event Listeners and Callback Methods .....	186
21.4 An Event Handling Example .....	187
21.5 Designing the User Interface .....	187
21.6 The Event Listener and Callback Method .....	189
21.7 Consuming Events .....	191
21.8 Summary.....	193
<b>22. A Guide to using Instant Run in Android Studio 2 .....</b>	<b>195</b>
22.1 Introducing Instant Run .....	195
22.2 Understanding Instant Run Swapping Levels .....	195
22.3 Enabling and Disabling Instant Run .....	196
22.4 Using Instant Run.....	197
22.5 An Instant Run Tutorial.....	197
22.6 Triggering an Instant Run Hot Swap .....	197
22.7 Triggering an Instant Run Warm Swap .....	198
22.8 Forcing a Warm Swap .....	199
22.9 Triggering an Instant Run Cold Swap .....	199
22.10 Making a Manifest Change .....	199
22.11 Summary.....	200
<b>23. Android Touch and Multi-touch Event Handling .....</b>	<b>201</b>
23.1 Intercepting Touch Events.....	201
23.2 The MotionEvent Object.....	202
23.3 Understanding Touch Actions.....	202
23.4 Handling Multiple Touches .....	202
23.5 An Example Multi-Touch Application .....	203

23.6 Designing the Activity User Interface .....	203
23.7 Implementing the Touch Event Listener .....	205
23.8 Running the Example Application .....	209
23.9 Summary .....	209
<b>24. Detecting Common Gestures using the Android Gesture Detector Class .....</b>	<b>211</b>
24.1 Implementing Common Gesture Detection .....	211
24.2 Creating an Example Gesture Detection Project .....	212
24.3 Implementing the Listener Class .....	213
24.4 Creating the GestureDetectorCompat Instance .....	215
24.5 Implementing the onTouchEvent() Method .....	216
24.6 Testing the Application .....	217
24.7 Summary .....	218
<b>25. Implementing Custom Gesture and Pinch Recognition on Android .....</b>	<b>219</b>
25.1 The Android Gesture Builder Application .....	219
25.2 The GestureOverlayView Class .....	219
25.3 Detecting Gestures .....	219
25.4 Identifying Specific Gestures .....	220
25.5 Building and Running the Gesture Builder Application .....	220
25.6 Creating a Gestures File .....	222
25.7 Extracting the Gestures File from the SD Card .....	224
25.8 Creating the Example Project .....	224
25.9 Adding the Gestures File to the Project .....	225
25.10 Designing the User Interface .....	225
25.11 Loading the Gestures File .....	226
25.12 Registering the Event Listener .....	227
25.13 Implementing the onGesturePerformed Method .....	227
25.14 Testing the Application .....	229
25.15 Configuring the GestureOverlayView .....	229
25.16 Intercepting Gestures .....	230
25.17 Detecting Pinch Gestures .....	230
25.18 A Pinch Gesture Example Project .....	231
25.19 Summary .....	233
<b>26. An Introduction to Android Fragments .....</b>	<b>235</b>
26.1 What is a Fragment? .....	235
26.2 Creating a Fragment .....	236
26.3 Adding a Fragment to an Activity using the Layout XML File .....	237
26.4 Adding and Managing Fragments in Code .....	239
26.5 Handling Fragment Events .....	240

26.6 Implementing Fragment Communication.....	241
26.7 Summary.....	243
<b>27. Using Fragments in Android Studio - An Example .....</b>	<b>245</b>
27.1 About the Example Fragment Application.....	245
27.2 Creating the Example Project .....	245
27.3 Creating the First Fragment Layout .....	246
27.4 Creating the First Fragment Class .....	248
27.5 Creating the Second Fragment Layout .....	249
27.6 Adding the Fragments to the Activity .....	251
27.7 Making the Toolbar Fragment Talk to the Activity .....	254
27.8 Making the Activity Talk to the Text Fragment .....	258
27.9 Testing the Application .....	259
27.10 Summary.....	260
<b>28. Creating and Managing Overflow Menus on Android .....</b>	<b>261</b>
28.1 The Overflow Menu .....	261
28.2 Creating an Overflow Menu .....	262
28.3 Displaying an Overflow Menu.....	263
28.4 Responding to Menu Item Selections.....	264
28.5 Creating Checkable Item Groups .....	264
28.6 Creating the Example Project .....	266
28.7 Modifying the Menu Description.....	266
28.8 Modifying the onOptionsItemSelected() Method .....	267
28.9 Testing the Application .....	269
28.10 Summary.....	269
<b>29. Animating User Interfaces with the Android Transitions Framework.....</b>	<b>271</b>
29.1 Introducing Android Transitions and Scenes .....	271
29.2 Using Interpolators with Transitions .....	272
29.3 Working with Scene Transitions .....	273
29.4 Custom Transitions and TransitionSets in Code .....	274
29.5 Custom Transitions and TransitionSets in XML .....	275
29.6 Working with Interpolators .....	277
29.7 Creating a Custom Interpolator .....	279
29.8 Using the beginDelayedTransition Method .....	280
29.9 Summary.....	280
<b>30. An Android Transition Tutorial using beginDelayedTransition .....</b>	<b>283</b>
30.1 Creating the Android Studio TransitionDemo Project .....	283
30.2 Preparing the Project Files.....	283
30.3 Implementing beginDelayedTransition Animation.....	284

30.4 Customizing the Transition.....	287
30.5 Summary .....	288
<b>31. Implementing Android Scene Transitions – A Tutorial .....</b>	<b>289</b>
31.1 An Overview of the Scene Transition Project.....	289
31.2 Creating the Android Studio SceneTransitions Project .....	289
31.3 Identifying and Preparing the Root Container .....	289
31.4 Designing the First Scene .....	290
31.5 Designing the Second Scene.....	292
31.6 Entering the First Scene.....	294
31.7 Loading Scene 2.....	295
31.8 Implementing the Transitions .....	296
31.9 Adding the Transition File.....	296
31.10 Loading and Using the Transition Set .....	297
31.11 Configuring Additional Transitions .....	298
31.12 Summary .....	299
<b>32. Working with the Floating Action Button and Snackbar .....</b>	<b>301</b>
32.1 The Material Design .....	301
32.2 The Design Library .....	302
32.3 The Floating Action Button (FAB) .....	302
32.4 The Snackbar .....	303
32.5 Creating the Example Project.....	303
32.6 Reviewing the Project.....	304
32.7 Changing the Floating Action Button .....	306
32.8 Adding the ListView to the Content Layout .....	308
32.9 Adding Items to the ListView.....	308
32.10 Adding an Action to the Snackbar .....	312
32.11 Summary .....	313
<b>33. Creating a Tabbed Interface using the TabLayout Component .....</b>	<b>315</b>
33.1 An Introduction to the ViewPager.....	315
33.2 An Overview of the TabLayout Component .....	315
33.3 Creating the TabLayoutDemo Project .....	316
33.4 Creating the First Fragment.....	316
33.5 Duplicating the Fragments .....	318
33.6 Adding the TabLayout and ViewPager .....	319
33.7 Creating the Pager Adapter .....	320
33.8 Performing the Initialization Tasks .....	322
33.9 Testing the Application.....	324
33.10 Customizing the TabLayout .....	325

33.11 Displaying Icon Tab Items .....	327
33.12 Summary.....	328
<b>34. Working with the RecyclerView and CardView Widgets .....</b>	<b>329</b>
34.1 An Overview of the RecyclerView.....	329
34.2 An Overview of the CardView.....	332
34.3 Adding the Libraries to the Project.....	333
34.4 Summary.....	334
<b>35. An Android RecyclerView and CardView Tutorial .....</b>	<b>335</b>
35.1 Creating the CardDemo Project.....	335
35.2 Removing the Floating Action Button.....	335
35.3 Adding the RecyclerView and CardView Libraries .....	336
35.4 Designing the CardView Layout.....	336
35.5 Adding the RecyclerView .....	338
35.6 Creating the RecyclerView Adapter.....	338
35.7 Adding the Image Files.....	341
35.8 Initializing the RecyclerView Component .....	342
35.9 Testing the Application .....	343
35.10 Responding to Card Selections .....	343
35.11 Summary.....	345
<b>36. Working with the AppBar and Collapsing Toolbar Layouts .....</b>	<b>347</b>
36.1 The Anatomy of an AppBar.....	347
36.2 The Example Project .....	348
36.3 Coordinating the RecyclerView and Toolbar .....	349
36.4 Introducing the Collapsing Toolbar Layout.....	351
36.5 Changing the Title and Scrim Color.....	354
36.6 Summary.....	355
<b>37. Implementing an Android Navigation Drawer .....</b>	<b>357</b>
37.1 An Overview of the Navigation Drawer .....	357
37.2 Opening and Closing the Drawer .....	359
37.3 Responding to Drawer Item Selections .....	359
37.4 Using the Navigation Drawer Activity Template.....	360
37.5 Creating the Navigation Drawer Template Project.....	361
37.6 The Template Layout Resource Files .....	361
37.7 The Header Coloring Resource File.....	361
37.8 The Template Menu Resource File .....	361
37.9 The Template Code.....	362
37.10 Running the App .....	363
37.11 Summary.....	364

<b>38. An Android Studio Master/Detail Flow Tutorial .....</b>	<b>365</b>
38.1 The Master/Detail Flow .....	365
38.2 Creating a Master/Detail Flow Activity .....	367
38.3 The Anatomy of the Master/Detail Flow Template.....	368
38.4 Modifying the Master/Detail Flow Template .....	369
38.5 Changing the Content Model .....	370
38.6 Changing the Detail Pane .....	372
38.7 Modifying the WebsiteDetailFragment Class .....	373
38.8 Modifying the WebsiteListActivity Class .....	374
38.9 Adding Manifest Permissions .....	375
38.10 Running the Application .....	375
38.11 Summary .....	376
<b>39. An Overview of Android Intents.....</b>	<b>377</b>
39.1 An Overview of Intents.....	377
39.2 Explicit Intents .....	378
39.3 Returning Data from an Activity .....	379
39.4 Implicit Intents .....	380
39.5 Using Intent Filters .....	381
39.6 Checking Intent Availability .....	382
39.7 Summary .....	382
<b>40. Android Explicit Intents – A Worked Example .....</b>	<b>385</b>
40.1 Creating the Explicit Intent Example Application .....	385
40.2 Designing the User Interface Layout for ActivityA .....	385
40.3 Creating the Second Activity Class .....	387
40.4 Designing the User Interface Layout for ActivityB.....	388
40.5 Reviewing the Application Manifest File .....	390
40.6 Creating the Intent .....	391
40.7 Extracting Intent Data .....	392
40.8 Launching ActivityB as a Sub-Activity .....	393
40.9 Returning Data from a Sub-Activity.....	394
40.10 Testing the Application.....	395
40.11 Summary .....	395
<b>41. Android Implicit Intents – A Worked Example .....</b>	<b>397</b>
41.1 Creating the Android Studio Implicit Intent Example Project .....	397
41.2 Designing the User Interface .....	397
41.3 Creating the Implicit Intent .....	399
41.4 Adding a Second Matching Activity.....	400
41.5 Adding the Web View to the UI.....	400

41.6	Obtaining the Intent URL .....	401
41.7	Modifying the MyWebView Project Manifest File.....	402
41.8	Installing the MyWebView Package on a Device .....	404
41.9	Testing the Application .....	404
41.10	Summary.....	405
<b>42.</b>	<b>Android Broadcast Intents and Broadcast Receivers.....</b>	<b>407</b>
42.1	An Overview of Broadcast Intents .....	407
42.2	An Overview of Broadcast Receivers .....	408
42.3	Obtaining Results from a Broadcast .....	410
42.4	Sticky Broadcast Intents .....	410
42.5	The Broadcast Intent Example.....	411
42.6	Creating the Example Application .....	411
42.7	Creating and Sending the Broadcast Intent.....	411
42.8	Creating the Broadcast Receiver .....	412
42.9	Configuring a Broadcast Receiver in the Manifest File .....	414
42.10	Testing the Broadcast Example.....	415
42.11	Listening for System Broadcasts .....	415
42.12	Summary.....	416
<b>43.</b>	<b>A Basic Overview of Threads and Thread Handlers.....</b>	<b>419</b>
43.1	An Overview of Threads .....	419
43.2	The Application Main Thread .....	419
43.3	Thread Handlers .....	419
43.4	A Basic Threading Example .....	420
43.5	Creating a New Thread .....	423
43.6	Implementing a Thread Handler.....	424
43.7	Passing a Message to the Handler.....	426
43.8	Summary.....	428
<b>44.</b>	<b>An Overview of Android Started and Bound Services .....</b>	<b>429</b>
44.1	Started Services .....	429
44.2	Intent Service.....	430
44.3	Bound Service .....	430
44.4	The Anatomy of a Service .....	431
44.5	Controlling Destroyed Service Restart Options .....	432
44.6	Declaring a Service in the Manifest File.....	432
44.7	Starting a Service Running on System Startup.....	433
44.8	Summary.....	433
<b>45.</b>	<b>Implementing an Android Started Service – A Worked Example.....</b>	<b>435</b>
45.1	Creating the Example Project .....	435



45.2 Creating the Service Class .....	435
45.3 Adding the Service to the Manifest File .....	437
45.4 Starting the Service .....	438
45.5 Testing the IntentService Example .....	438
45.6 Using the Service Class .....	439
45.7 Creating the New Service .....	439
45.8 Modifying the User Interface .....	441
45.9 Running the Application .....	443
45.10 Creating a New Thread for Service Tasks .....	443
45.11 Summary .....	445
<b>46. Android Local Bound Services – A Worked Example.....</b>	<b>447</b>
46.1 Understanding Bound Services .....	447
46.2 Bound Service Interaction Options.....	447
46.3 An Android Studio Local Bound Service Example.....	448
46.4 Adding a Bound Service to the Project.....	448
46.5 Implementing the Binder .....	449
46.6 Binding the Client to the Service .....	452
46.7 Completing the Example .....	453
46.8 Testing the Application.....	456
46.9 Summary .....	456
<b>47. Android Remote Bound Services – A Worked Example .....</b>	<b>457</b>
47.1 Client to Remote Service Communication.....	457
47.2 Creating the Example Application .....	457
47.3 Designing the User Interface .....	458
47.4 Implementing the Remote Bound Service .....	458
47.5 Configuring a Remote Service in the Manifest File .....	460
47.6 Launching and Binding to the Remote Service .....	461
47.7 Sending a Message to the Remote Service .....	463
47.8 Summary .....	463
<b>48. An Overview of Android SQLite Databases .....</b>	<b>465</b>
48.1 Understanding Database Tables.....	465
48.2 Introducing Database Schema.....	466
48.3 Columns and Data Types .....	466
48.4 Database Rows .....	466
48.5 Introducing Primary Keys .....	466
48.6 What is SQLite? .....	467
48.7 Structured Query Language (SQL) .....	467
48.8 Trying SQLite on an Android Virtual Device (AVD) .....	468

48.9 Android SQLite Java Classes.....	470
48.9.1 <i>Cursor</i> .....	470
48.9.2 <i>SQLiteDatabase</i> .....	471
48.9.3 <i>SQLiteOpenHelper</i> .....	471
48.9.4 <i>ContentValues</i> .....	472
48.10 Summary.....	472
<b>49. An Android TableLayout and TableRow Tutorial.....</b>	<b>473</b>
49.1 The TableLayout and TableRow Layout Views.....	473
49.2 Creating the Database Project.....	475
49.3 Adding the TableLayout to the User Interface .....	475
49.4 Adding and Configuring the TableRows.....	476
49.5 Adding the Button Bar to the Layout.....	477
49.6 Adjusting the Layout Margins.....	479
49.7 Summary.....	481
<b>50. An Android SQLite Database Tutorial .....</b>	<b>483</b>
50.1 About the Database Example .....	483
50.2 Creating the Data Model .....	484
50.3 Implementing the Data Handler .....	485
50.3.1 <i>The Add Handler Method</i> .....	488
50.3.2 <i>The Query Handler Method</i> .....	488
50.3.3 <i>The Delete Handler Method</i> .....	489
50.4 Implementing the Activity Event Methods.....	489
50.5 Testing the Application .....	492
50.6 Summary.....	492
<b>51. Understanding Android Content Providers .....</b>	<b>493</b>
51.1 What is a Content Provider?.....	493
51.2 The Content Provider.....	493
51.2.1 <i>onCreate()</i> .....	494
51.2.2 <i>query()</i> .....	494
51.2.3 <i>insert()</i> .....	494
51.2.4 <i>update()</i> .....	494
51.2.5 <i>delete()</i> .....	494
51.2.6 <i>getType()</i> .....	494
51.3 The Content URI.....	494
51.4 The Content Resolver .....	495
51.5 The <provider> Manifest Element .....	495
51.6 Summary.....	496
<b>52. Implementing an Android Content Provider in Android Studio.....</b>	<b>497</b>

52.1 Copying the Database Project .....	497
52.2 Adding the Content Provider Package.....	497
52.3 Creating the Content Provider Class .....	498
52.4 Constructing the Authority and Content URI .....	500
52.5 Implementing URI Matching in the Content Provider.....	501
52.6 Implementing the Content Provider onCreate() Method .....	503
52.7 Implementing the Content Provider insert() Method .....	503
52.8 Implementing the Content Provider query() Method .....	504
52.9 Implementing the Content Provider update() Method .....	506
52.10 Implementing the Content Provider delete() Method .....	507
52.11 Declaring the Content Provider in the Manifest File .....	509
52.12 Modifying the Database Handler .....	510
52.13 Summary .....	512
<b>53. Accessing Cloud Storage using the Android Storage Access Framework.....</b>	<b>513</b>
53.1 The Storage Access Framework.....	513
53.2 Working with the Storage Access Framework.....	515
53.3 Filtering Picker File Listings.....	515
53.4 Handling Intent Results .....	517
53.5 Reading the Content of a File .....	517
53.6 Writing Content to a File .....	518
53.7 Deleting a File .....	519
53.8 Gaining Persistent Access to a File .....	519
53.9 Summary .....	520
<b>54. An Android Storage Access Framework Example .....</b>	<b>521</b>
54.1 About the Storage Access Framework Example .....	521
54.2 Creating the Storage Access Framework Example .....	521
54.3 Designing the User Interface .....	522
54.4 Declaring Request Codes .....	524
54.5 Creating a New Storage File .....	525
54.6 The onActivityResult() Method .....	526
54.7 Saving to a Storage File .....	528
54.8 Opening and Reading a Storage File.....	531
54.9 Testing the Storage Access Application.....	533
54.10 Summary .....	534
<b>55. Implementing Video Playback on Android using the VideoView and MediaController Classes .....</b>	<b>535</b>
55.1 Introducing the Android VideoView Class .....	535
55.2 Introducing the Android MediaController Class.....	536
55.3 Testing Video Playback.....	537

55.4	Creating the Video Playback Example .....	537
55.5	Designing the VideoPlayer Layout .....	537
55.6	Configuring the VideoView .....	538
55.7	Adding Internet Permission .....	539
55.8	Adding the MediaController to the Video View .....	540
55.9	Setting up the onPreparedListener .....	541
55.10	Summary .....	543
<b>56.</b>	<b>Video Recording and Image Capture on Android using Camera Intents .....</b>	<b>545</b>
56.1	Checking for Camera Support .....	545
56.2	Calling the Video Capture Intent .....	546
56.3	Calling the Image Capture Intent .....	547
56.4	Creating an Android Studio Video Recording Project .....	548
56.5	Designing the User Interface Layout .....	548
56.6	Checking for the Camera .....	549
56.7	Launching the Video Capture Intent .....	550
56.8	Handling the Intent Return .....	551
56.9	Testing the Application .....	552
56.10	Summary .....	552
<b>57.</b>	<b>Making Runtime Permission Requests in Android 6.0 .....</b>	<b>553</b>
57.1	Understanding Normal and Dangerous Permissions .....	553
57.2	Creating the Permissions Example Project .....	555
57.3	Checking for a Permission .....	555
57.4	Requesting Permission at Runtime .....	557
57.5	Providing a Rationale for the Permission Request .....	559
57.6	Testing the Permissions App .....	561
57.7	Summary .....	562
<b>58.</b>	<b>Android Audio Recording and Playback using MediaPlayer and MediaRecorder .....</b>	<b>563</b>
58.1	Playing Audio .....	563
58.2	Recording Audio and Video using the MediaRecorder Class .....	564
58.3	About the Example Project .....	565
58.4	Creating the AudioApp Project .....	566
58.5	Designing the User Interface .....	566
58.6	Checking for Microphone Availability .....	567
58.7	Performing the Activity Initialization .....	568
58.8	Implementing the recordAudio() Method .....	569
58.9	Implementing the stopAudio() Method .....	570
58.10	Implementing the playAudio() method .....	571
58.11	Configuring and Requesting Permissions .....	571

58.12 Testing the Application.....	575
58.13 Summary .....	575
<b>59. Working with the Google Maps Android API in Android Studio .....</b>	<b>577</b>
59.1 The Elements of the Google Maps Android API .....	577
59.2 Creating the Google Maps Project .....	578
59.3 Obtaining Your Developer Signature.....	578
59.4 Testing the Application.....	580
59.5 Understanding Geocoding and Reverse Geocoding .....	580
59.6 Adding a Map to an Application .....	582
59.7 Requesting Current Location Permission .....	583
59.8 Displaying the User’s Current Location .....	585
59.9 Changing the Map Type.....	586
59.10 Displaying Map Controls to the User.....	587
59.11 Handling Map Gesture Interaction .....	588
59.11.1 Map Zooming Gestures.....	588
59.11.2 Map Scrolling/Panning Gestures.....	588
59.11.3 Map Tilt Gestures.....	589
59.11.4 Map Rotation Gestures .....	589
59.12 Creating Map Markers.....	589
59.13 Controlling the Map Camera .....	590
59.14 Summary .....	592
<b>60. Printing with the Android Printing Framework .....</b>	<b>593</b>
60.1 The Android Printing Architecture .....	593
60.2 The Print Service Plugins .....	593
60.3 Google Cloud Print .....	594
60.4 Printing to Google Drive .....	595
60.5 Save as PDF.....	595
60.6 Printing from Android Devices .....	595
60.7 Options for Building Print Support into Android Apps .....	597
60.7.1 Image Printing.....	597
60.7.2 Creating and Printing HTML Content .....	598
60.7.3 Printing a Web Page .....	600
60.7.4 Printing a Custom Document .....	601
60.8 Summary .....	601
<b>61. An Android HTML and Web Content Printing Example .....</b>	<b>603</b>
61.1 Creating the HTML Printing Example Application .....	603
61.2 Printing Dynamic HTML Content .....	603
61.3 Creating the Web Page Printing Example.....	607

61.4 Removing the Floating Action Button.....	607
61.5 Designing the User Interface Layout .....	608
61.6 Loading the Web Page into the WebView .....	610
61.7 Adding the Print Menu Option .....	611
61.8 Summary.....	614
<b>62. A Guide to Android Custom Document Printing.....</b>	<b>615</b>
62.1 An Overview of Android Custom Document Printing .....	615
62.1.1 Custom Print Adapters .....	616
62.2 Preparing the Custom Document Printing Project .....	617
62.3 Creating the Custom Print Adapter .....	618
62.4 Implementing the onLayout() Callback Method.....	619
62.5 Implementing the onWrite() Callback Method.....	623
62.6 Checking a Page is in Range.....	626
62.7 Drawing the Content on the Page Canvas .....	627
62.8 Starting the Print Job .....	630
62.9 Testing the Application .....	631
62.10 Summary.....	632
<b>63. An Android Fingerprint Authentication Tutorial .....</b>	<b>633</b>
63.1 An Overview of Fingerprint Authentication.....	633
63.2 Creating the Fingerprint Authentication Project .....	634
63.3 Configuring Device Fingerprint Authentication .....	634
63.4 Adding the Fingerprint Permission to the Manifest File.....	635
63.5 Downloading the Fingerprint Icon .....	635
63.6 Designing the User Interface .....	636
63.7 Accessing the Keyguard and Fingerprint Manager Services .....	638
63.8 Checking the Security Settings.....	638
63.9 Accessing the Android Keystore and KeyGenerator .....	640
63.10 Generating the Key .....	642
63.11 Initializing the Cipher .....	644
63.12 Creating the CryptoObject Instance .....	646
63.13 Implementing the Fingerprint Authentication Handler Class.....	647
63.14 Testing the Project.....	650
63.15 Summary.....	650
<b>64. Handling Different Android Devices and Displays .....</b>	<b>653</b>
64.1 Handling Different Device Displays.....	653
64.2 Creating a Layout for each Display Size .....	653
64.3 Providing Different Images .....	654
64.4 Checking for Hardware Support .....	655

64.5 Providing Device Specific Application Binaries .....	656
64.6 Summary .....	656
<b>65. Signing and Preparing an Android Application for Release .....</b>	<b>657</b>
65.1 The Release Preparation Process .....	657
65.2 Changing the Build Variant .....	657
65.3 Enabling ProGuard.....	658
65.4 Creating a Keystore File .....	659
65.5 Generating a Private Key .....	660
65.6 Creating the Application APK File .....	661
65.7 Register for a Google Play Developer Console Account .....	663
65.8 Uploading New APK Versions to the Google Play Developer Console .....	663
65.9 Summary .....	665
<b>66. Integrating Google Play In-app Billing into an Android Application .....</b>	<b>667</b>
66.1 Installing the Google Play Billing Library .....	667
66.2 Creating the Example In-app Billing Project .....	668
66.3 Adding Billing Permission to the Manifest File .....	669
66.4 Adding the InAppBillingService.aidl File to the Project .....	669
66.5 Adding the Utility Classes to the Project .....	671
66.6 Designing the User Interface .....	672
66.7 Implementing the “Click Me” Button .....	674
66.8 Google Play Developer Console and Google Wallet Accounts .....	675
66.9 Obtaining the Public License Key for the Application.....	675
66.10 Setting Up Google Play Billing in the Application .....	676
66.11 Initiating a Google Play In-app Billing Purchase .....	678
66.12 Implementing the onActivityResult Method .....	679
66.13 Implementing the Purchase Finished Listener .....	680
66.14 Consuming the Purchased Item .....	681
66.15 Releasing the IabHelper Instance .....	682
66.16 Modifying the Security.java File .....	682
66.17 Testing the In-app Billing Application.....	684
66.18 Building a Release APK .....	684
66.19 Creating a New In-app Product .....	685
66.20 Publishing the Application to the Alpha Distribution Channel .....	686
66.21 Adding In-app Billing Test Accounts .....	687
66.22 Configuring Group Testing.....	688
66.23 Resolving Problems with In-App Purchasing .....	689
66.24 Summary .....	690
<b>67. An Overview of Gradle in Android Studio .....</b>	<b>691</b>

## Introduction

67.1 An Overview of Gradle.....	691
67.2 Gradle and Android Studio .....	691
67.2.1 <i>Sensible Defaults</i> .....	692
67.2.2 <i>Dependencies</i> .....	692
67.2.3 <i>Build Variants</i> .....	692
67.2.4 <i>Manifest Entries</i> .....	693
67.2.5 <i>APK Signing</i> .....	693
67.2.6 <i>ProGuard Support</i> .....	693
67.3 The Top-level Gradle Build File .....	693
67.4 Module Level Gradle Build Files .....	695
67.5 Configuring Signing Settings in the Build File .....	698
67.6 Running Gradle Tasks from the Command-line.....	699
67.7 Summary.....	700
<b>68. An Android Studio Gradle Build Variants Example .....</b>	<b>701</b>
68.1 Creating the Build Variant Example Project .....	701
68.2 Extracting the Hello World String Resource .....	702
68.3 Adding the Build Flavors to the Module Build File .....	702
68.4 Adding the Flavors to the Project Structure .....	705
68.5 Adding Resource Files to the Flavors.....	706
68.6 Testing the Build Flavors.....	708
68.7 Build Variants and Class Files.....	708
68.8 Adding Packages to the Build Flavors .....	708
68.9 Customizing the Activity Classes.....	709
68.10 Summary.....	710
<b>Index .....</b>	<b>711</b>



# 1. Introduction

**F**ully updated for Android Studio 2, the goal of this book is to teach the skills necessary to develop Android based applications using the Android Studio Integrated Development Environment (IDE) and the Android 6 Software Development Kit (SDK).

Beginning with the basics, this book provides an outline of the steps necessary to set up an Android development and testing environment. An overview of Android Studio is included covering areas such as tool windows, the code editor and the Designer tool. An introduction to the architecture of Android is followed by an in-depth look at the design of Android applications and user interfaces using the Android Studio environment. More advanced topics such as database management, content providers and intents are also covered, as are touch screen handling, gesture recognition, camera access and the playback and recording of both video and audio. This edition of the book also covers printing, transitions and cloud-based file storage.

The concepts of material design are also covered in detail, including the use of floating action buttons, Snackbars, tabbed interfaces, card views, navigation drawers and collapsing toolbars.

In addition to covering general Android development techniques, the book also includes Google Play specific topics such as implementing maps using the Google Maps Android API, in-app billing and submitting apps to the Google Play Developer Console.

The key new features of Android Studio 2, Instant Run and the new AVD emulator environment, are also covered in detail.

Chapters also cover advanced features of Android Studio such as Gradle build configuration and the implementation of build variants to target multiple Android device types from a single project code base.

Assuming you already have some Java programming experience, are ready to download Android Studio and the Android SDK, have access to a Windows, Mac or Linux system and ideas for some apps to develop, you are ready to get started.

## 1.1 Downloading the Code Samples

The source code and Android Studio project files for the examples contained in this book are available for download at:

<http://www.ebookfrenzy.com/print/androidstudio2/index.php>

The steps to load a project from the code samples into Android Studio are as follows:

1. From the *Welcome to Android Studio* dialog, select the *Open an existing Android Studio* project option.
2. In the project selection dialog, navigate to and select the folder containing the project to be imported and click on OK.

## 1.2 Download the eBook

Thank you for purchasing the print edition of this book. If you would like to download the eBook version of this book, please email proof of purchase to [feedback@ebookfrenzy.com](mailto:feedback@ebookfrenzy.com) and we will provide you with a download link for the book in PDF, ePub and MOBI formats.

## 1.3 Feedback

We want you to be satisfied with your purchase of this book. If you find any errors in the book, or have any comments, questions or concerns please contact us at [feedback@ebookfrenzy.com](mailto:feedback@ebookfrenzy.com).

## 1.4 Errata

While we make every effort to ensure the accuracy of the content of this book, it is inevitable that a book covering a subject area of this size and complexity may include some errors and oversights. Any known issues with the book will be outlined, together with solutions, at the following URL:

<http://www.ebookfrenzy.com/errata/androidstudio2.html>

In the event that you find an error not listed in the errata, please let us know by emailing our technical support team at [feedback@ebookfrenzy.com](mailto:feedback@ebookfrenzy.com). They are there to help you and will work to resolve any problems you may encounter.

## 2. Setting up an Android Studio Development Environment

**B**efore any work can begin on the development of an Android application, the first step is to configure a computer system to act as the development platform. This involves a number of steps consisting of installing the Java Development Kit (JDK) and the Android Studio Integrated Development Environment (IDE) which also includes the Android Software Development Kit (SDK).

This chapter will cover the steps necessary to install the requisite components for Android application development on Windows, Mac OS X and Linux based systems.

### 2.1 System Requirements

Android application development may be performed on any of the following system types:

- Windows 7/8/10 (32-bit or 64-bit)
- Mac OS X 10.8.5 or later (Intel based systems only)
- Linux systems with version 2.11 or later of GNU C Library (glibc)
- Minimum of 2GB of RAM (8GB is preferred)
- Approximately 4GB of available disk space
- 1280 x 800 minimum screen resolution

### 2.2 Installing the Java Development Kit (JDK)

The Android SDK was developed using the Java programming language. Similarly, Android applications are also developed using Java. As a result, the Java Development Kit (JDK) is the first component that must be installed.

Android Studio 2 development requires the installation of version 8 of the Standard Edition of the Java Platform Development Kit. Java is provided in both development (JDK) and runtime (JRE) packages. For the purposes of Android development, the JDK must be installed.

## 2.2.1 Windows JDK Installation

For Windows systems, the JDK may be obtained from Oracle Corporation's website using the following URL:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Assuming that a suitable JDK is not already installed on your system, download version 8 of the JDK package that matches the destination computer system. Once downloaded, launch the installation executable and follow the on screen instructions to complete the installation process.

## 2.2.2 Mac OS X JDK Installation

Java is not installed by default on recent versions of Mac OS X. To confirm the presence or otherwise of Java, open a Terminal window and enter the following command:

```
java -version
```

Assuming that Java is currently installed, output similar to the following will appear in the terminal window:

```
java version "1.7.0_79-b15"  
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)  
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
```

In the event that Java is not installed, issuing the "java" command in the terminal window will result in the appearance of a message which reads as follows together with a dialog on the desktop providing a More Info button which, when clicked will display the Oracle Java web page:

```
No Java runtime present, requesting install
```

On the Oracle Java web page, locate and download the Java SE 8 JDK installation package for Mac OS X.

Open the downloaded disk image (.dmg file) and double-click on the icon to install the Java package (Figure 2-1):



Figure 2-1

The Java for OS X installer window will appear and take you through the steps involved in installing the JDK. Once the installation is complete, return to the Terminal window and run the following command, at which point the previously outlined Java version information should appear:

```
java -version
```

## 2.3 Linux JDK Installation

First, if the chosen development system is running the 64-bit version of Ubuntu then it is essential that a 32-bit library support package be installed:

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

As with Windows based JDK installation, it is possible to install the JDK on Linux by downloading the appropriate package from the Oracle web site, the URL for which is as follows:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Packages are provided by Oracle in RPM format (for installation on Red Hat Linux based systems such as Red Hat Enterprise Linux, Fedora and CentOS) and as a tar archive for other Linux distributions such as Ubuntu.

On Red Hat based Linux systems, download the .rpm JDK file from the Oracle web site and perform the installation using the *rpm* command in a terminal window. Assuming, for example, that the downloaded JDK file was named *jdk-8u77-linux-x64.rpm*, the commands to perform the installation would read as follows:

```
su
rpm -ihv jdk-8u77-linux-x64.rpm
```

## Setting up an Android Studio Development Environment

To install using the compressed tar package (tar.gz) perform the following steps:

1. Create the directory into which the JDK is to be installed (for the purposes of this example we will assume */home/demo/java*).
2. Download the appropriate tar.gz package from the Oracle web site into the directory.
3. Execute the following command (where *<jdk-file>* is replaced by the name of the downloaded JDK file):

```
tar xvfz <jdk-file>.tar.gz
```

4. Remove the downloaded tar.gz file.
5. Add the path to the *bin* directory of the JDK installation to your \$PATH variable. For example, assuming that the JDK ultimately installed into */home/demo/java/jdk1.8.0\_77* the following would need to be added to your \$PATH environment variable:

```
/home/demo/java/jdk1.8.0_77/bin
```

This can typically be achieved by adding a command to the *.bashrc* file in your home directory (specifics may differ depending on the particular Linux distribution in use). For example, change directory to your home directory, edit the *.bashrc* file contained therein and add the following line at the end of the file (modifying the path to match the location of the JDK on your system):

```
export PATH=/home/demo/java/jdk1.8.0_77/bin:$PATH
```

Having saved the change, future terminal sessions will include the JDK in the \$PATH environment variable.

## 2.4 Downloading the Android Studio Package

Most of the work involved in developing applications for Android will be performed using the Android Studio environment. The content and examples in this book were created based on Android Studio version 2.0.

Android Studio is subject to frequent updates and it is possible, therefore, that a more recent release of Android Studio is now available. For the purposes of compatibility with the tutorials and examples, however, it is recommended that this book be used with Android Studio version 2.0 which may be downloaded from the following web page:

<http://tools.android.com/download/studio/builds/2-0>

From this page, select the appropriate package for your platform and operating system. On the subsequent screen, accept the terms and conditions to initiate the download.

## 2.5 Installing Android Studio

Once downloaded, the exact steps to install Android Studio differ depending on the operating system on which the installation is being performed.

### 2.5.1 Installation on Windows

Locate the downloaded Android Studio installation executable file (named *android-studio-bundle-<version>.exe*) in a Windows Explorer window and double click on it to start the installation process, clicking the *Yes* button in the User Account Control dialog if it appears.

Once the Android Studio setup wizard appears, work through the various screens to configure the installation to meet your requirements in terms of the file system location into which Android Studio should be installed and whether or not it should be made available to other users of the system. When prompted to select the components to install, make sure that the *Android Studio*, *Android SDK* and *Android Virtual Device* options are all selected.

Although there are no strict rules on where Android Studio should be installed on the system, the remainder of this book will assume that the installation was performed into *C:\Program Files\Android\Android Studio* and that the Android SDK packages have been installed into the user's *AppData\Local\Android\sdk* sub-folder. Once the options have been configured, click on the *Install* button to begin the installation process.

On versions of Windows with a Start menu, the newly installed Android Studio can be launched from the entry added to that menu during the installation. The executable may be pinned to the task bar for easy access by navigating to the *Android Studio\bin* directory, right-clicking on the executable and selecting the *Pin to Taskbar* menu option. Note that the executable is provided in 32-bit (*studio*) and 64-bit (*studio64*) executable versions. If you are running a 32-bit system be sure to use the *studio* executable.

### 2.5.2 Installation on Mac OS X

Android Studio for Mac OS X is downloaded in the form of a disk image (.dmg) file. Once the *android-studio-ide-<version>.dmg* file has been downloaded, locate it in a Finder window and double click on it to open it as shown in Figure 2-2:



Figure 2-2

To install the package, simply drag the Android Studio icon and drop it onto the Applications folder. The Android Studio package will then be installed into the Applications folder of the system, a process which will typically take a few minutes to complete.

To launch Android Studio, locate the executable in the Applications folder using a Finder window and double click on it. When attempting to launch Android Studio, an error dialog may appear indicating that the JVM cannot be found. If this error occurs, it will be necessary to download and install the Mac OS X Java 6 JRE package on the system. This can be downloaded from Apple using the following link:

<http://support.apple.com/kb/DL1572>

Once the Java for OS X package has been installed, Android Studio should launch without any problems.

For future easier access to the tool, drag the Android Studio icon from the Finder window and drop it onto the dock.

### 2.5.3 Installation on Linux

Having downloaded the Linux Android Studio package, open a terminal window, change directory to the location where Android Studio is to be installed and execute the following command:

```
unzip /<path to package>/android-studio-ide-<version>-linux.zip
```

Note that the Android Studio bundle will be installed into a sub-directory named *android-studio*. Assuming, therefore, that the above command was executed in */home/demo*, the software packages will be unpacked into */home/demo/android-studio*.



To launch Android Studio, open a terminal window, change directory to the *android-studio/bin* sub-directory and execute the following command:

```
./studio.sh
```

On Linux it may also be necessary to specify the location of the Java Development Kit using the following steps:

1. Launch Android Studio and create a new project.
2. Select the *File -> Other Settings -> Default Project Structure...* menu option.
3. Enter the full path to the directory containing the JDK into the *JDK Location* field.
4. Click *Apply* followed by *OK*.

## 2.6 The Android Studio Setup Wizard

The first time that Android Studio is launched after being installed, a dialog will appear providing the option to import settings from a previous Android Studio version. If you have settings from a previous version and would like to import them into the latest installation, select the appropriate option and location. Alternatively, indicate that you do not need to import any previous settings and click on the OK button to proceed.

Next, the setup wizard may appear as shown in Figure 2-3 though this dialog does not appear on all platforms:

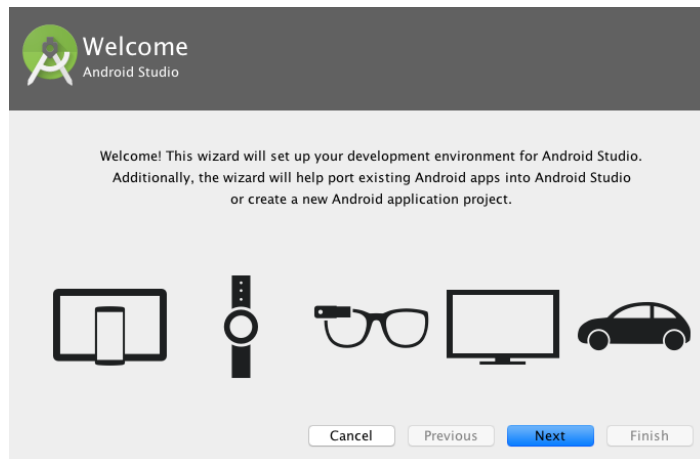


Figure 2-3

If the wizard appears, click on the Next button, choose the Standard installation option and click on Next once again.

## Setting up an Android Studio Development Environment

Android Studio will proceed to download and configure the latest Android SDK and some additional components and packages. Once this process has completed, click on the *Finish* button in the *Downloading Components* dialog at which point the Welcome to Android Studio screen should then appear:

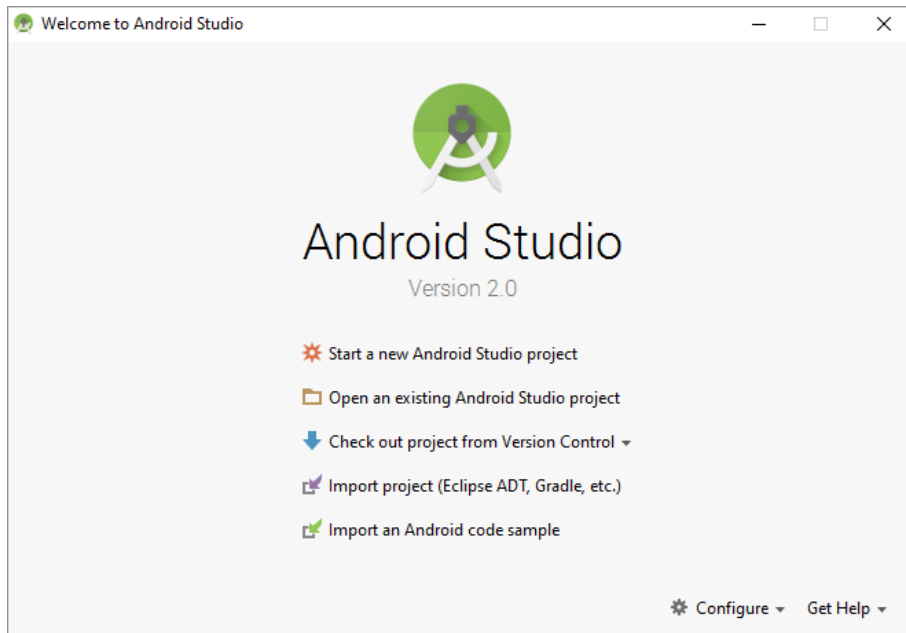


Figure 2-4

## 2.7 Installing Additional Android SDK Packages

The steps performed so far have installed Java, the Android Studio IDE and the current set of default Android SDK packages. Before proceeding, it is worth taking some time to verify which packages are installed and to install any missing or updated packages.

This task can be performed using the *Android SDK Settings* screen, which may be launched from within the Android Studio tool by selecting the *Configure* -> *SDK Manager* option from within the Android Studio welcome dialog. Once invoked, the *Android SDK* screen of the default settings dialog will appear as shown in Figure 2-5:

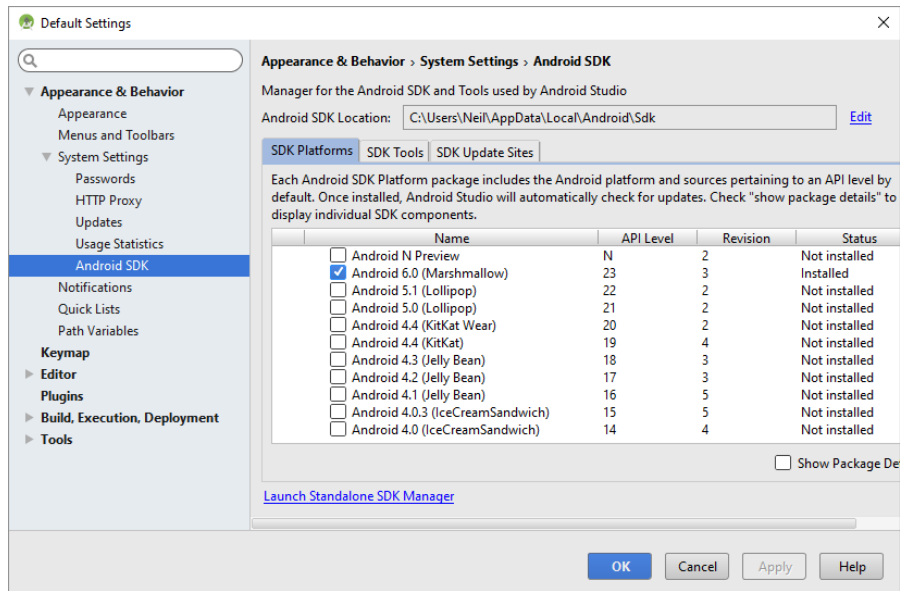


Figure 2-5

Immediately after installing Android Studio for the first time it is likely that only the latest version of the Android SDK has been installed. To install older versions of the Android SDK simply select the checkboxes corresponding to the versions and click on the *Apply* button.

It is also possible that updates will be listed as being available for the latest SDK. To access detailed information about the packages that are available for update, enable the *Show Package Details* option located in the lower right hand corner of the screen. This will display information similar to that shown in Figure 2-6:

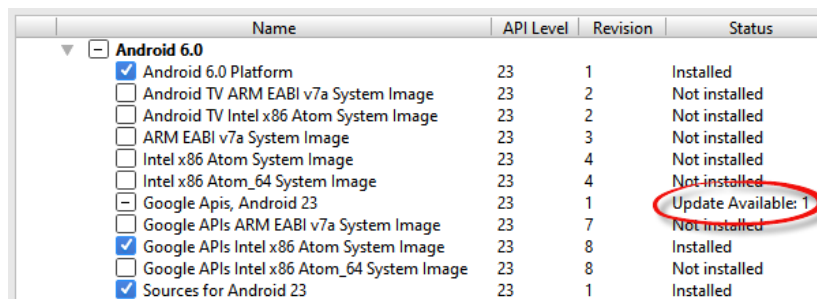


Figure 2-6

The above figure highlights the availability of an update. To install the updates, enable the checkbox to the left of the item name and click on the *Apply* button.

## Setting up an Android Studio Development Environment

In addition to the Android SDK packages, a number of tools are also installed for building Android applications. To view the currently installed packages and check for updates, remain within the SDK settings screen and select the SDK Tools tab as shown in Figure 2-7:

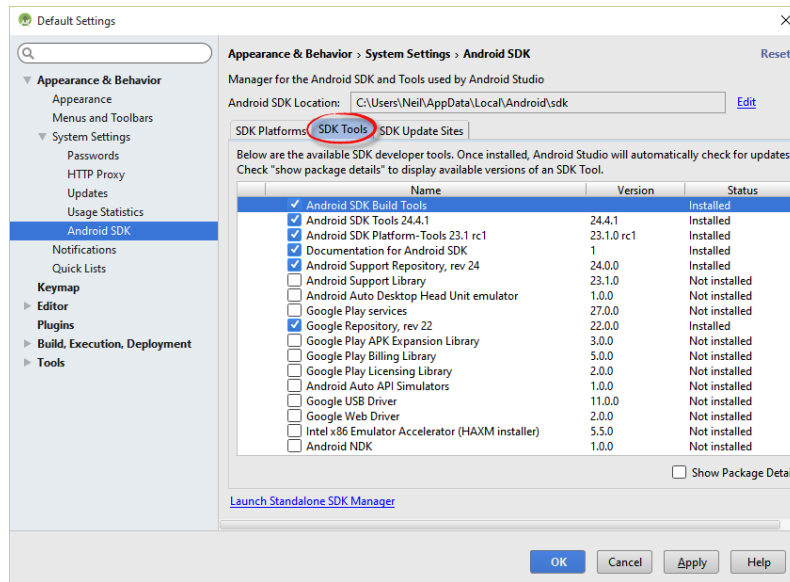


Figure 2-7

Within the Android SDK Tools screen, make sure that the following packages are listed as *Installed* in the Status column:

- Android SDK Build-tools
- Android SDK Tools
- Android SDK Platform-tools
- Android Support Repository
- Android Support Library
- Google Repository
- Google USB Driver (Windows only)
- Intel x86 Emulator Accelerator (HAXM installer)

In the event that any of the above packages are listed as *Not Installed* or requiring an update, simply select the checkboxes next to those packages and click on the *Apply* button to initiate the installation process.

Once the installation is complete, review the package list and make sure that the selected packages are now listed as *Installed* in the *Status* column. If any are listed as *Not installed*, make sure they are selected and click on the *Install packages...* button again.

An alternative to using the Android SDK settings panel is to access the *Standalone SDK Manager* which can be launched using the link in the lower left hand corner of the settings screen. The Standalone SDK Manager (Figure 2-8) provides a similar list of packages together with options to perform update and installation tasks:

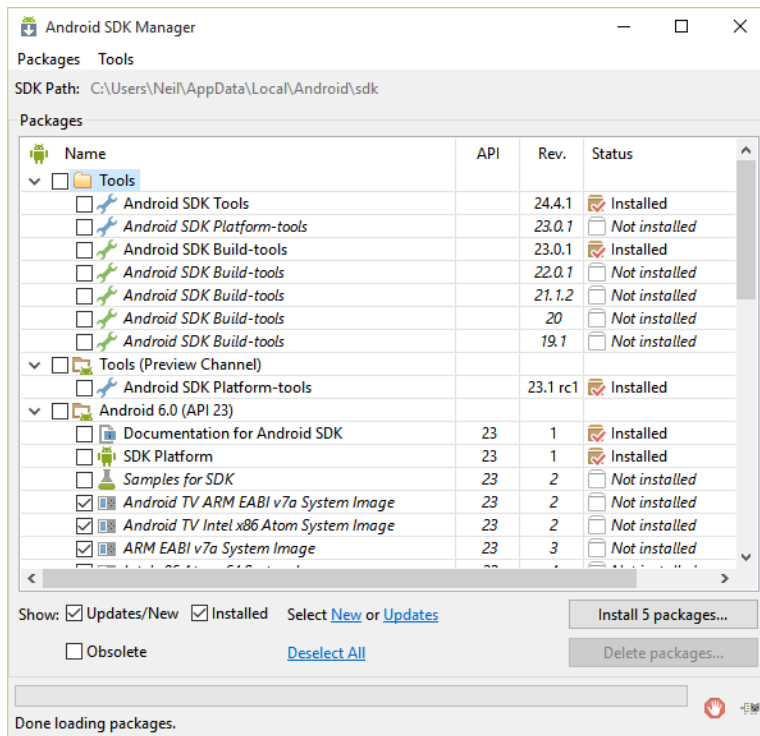


Figure 2-8

## 2.8 Making the Android SDK Tools Command-line Accessible

Most of the time, the underlying tools of the Android SDK will be accessed from within the Android Studio environment. That being said, however, there will also be instances where it will be useful to be able to invoke those tools from a command prompt or terminal window. In order for the operating system on which you are developing to be able to find these tools, it will be necessary to add them to the system's *PATH* environment variable.

Regardless of operating system, the *PATH* variable needs to be configured to include the following paths (where *<path\_to\_android\_sdk\_installation>* represents the file system location into which the Android SDK was installed):

```
<path_to_android_sdk_installation>/sdk/tools
<path_to_android_sdk_installation>/sdk/platform-tools
```

## Setting up an Android Studio Development Environment

The location of the SDK on your system can be identified by launching the Standalone SDK Manager and referring to the *Android SDK Location*: field located at the top of the settings panel as highlighted in Figure 2-9:

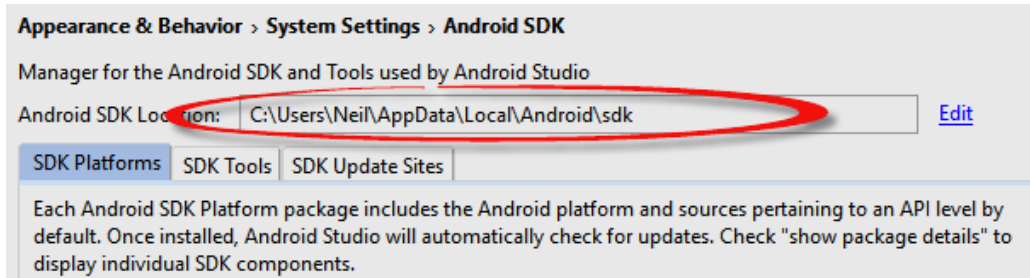


Figure 2-9

Once the location of the SDK has been identified, the steps to add this to the PATH variable are operating system dependent:

### 2.8.1 Windows 7

1. Right-click on *Computer* in the desktop start menu and select *Properties* from the resulting menu.
2. In the properties panel, select the *Advanced System Settings* link and, in the resulting dialog, click on the *Environment Variables...* button.
3. In the Environment Variables dialog, locate the *Path* variable in the *System variables* list, select it and click on *Edit...* Locate the end of the current variable value string and append the path to the Android platform tools to the end, using a semicolon to separate the path from the preceding values. For example, assuming the Android SDK was installed into *C:\Users\demo\AppData\Local\Android\sdk*, the following would be appended to the end of the current Path value:

```
;C:\Users\demo\AppData\Local\Android\sdk\platform-  
tools;C:\Users\demo\AppData\Local\Android\sdk\tools
```

4. Click on OK in each dialog box and close the system properties control panel.

Once the above steps are complete, verify that the path is correctly set by opening a *Command Prompt* window (*Start -> All Programs -> Accessories -> Command Prompt*) and at the prompt enter:

```
echo %Path%
```

The returned path variable value should include the paths to the Android SDK platform tools folders. Verify that the *platform-tools* value is correct by attempting to run the *adb* tool as follows:

```
adb
```

The tool should output a list of command line options when executed.

Similarly, check the *tools* path setting by attempting to launch the Android SDK Manager:

```
android
```

In the event that a message similar to the following message appears for one or both of the commands, it is most likely that an incorrect path was appended to the Path environment variable:

```
'adb' is not recognized as an internal or external command,  
operable program or batch file.
```

### 2.8.2 Windows 8.1

1. On the start screen, move the mouse to the bottom right hand corner of the screen and select *Search* from the resulting menu. In the search box, enter *Control Panel*. When the Control Panel icon appears in the results area, click on it to launch the tool on the desktop.
2. Within the Control Panel, use the *Category* menu to change the display to *Large Icons*. From the list of icons select the one labeled *System*.
3. Follow the steps outlined for Windows 7 starting from step 2 through to step 4.

Open the command prompt window (move the mouse to the bottom right hand corner of the screen, select the Search option and enter *cmd* into the search box). Select *Command Prompt* from the search results.

Within the Command Prompt window, enter:

```
echo %Path%
```

The returned path variable value should include the paths to the Android SDK platform tools folders. Verify that the *platform-tools* value is correct by attempting to run the *adb* tool as follows:

```
adb
```

The tool should output a list of command line options when executed.

Similarly, check the *tools* path setting by attempting to launch the Android SDK Manager:

```
android
```

In the event that a message similar to the following message appears for one or both of the commands, it is most likely that an incorrect path was appended to the Path environment variable:

```
'adb' is not recognized as an internal or external command,  
operable program or batch file.
```

### 2.8.3 Windows 10

Right-click on the Start menu, select *System* from the resulting menu and click on the *Advanced system settings* option in the System window. Follow the steps outlined for Windows 7 starting from step 2 through to step 4.

### 2.8.4 Linux

On Linux this will involve once again editing the `.bashrc` file. Assuming that the Android SDK bundle package was installed into `/home/demo/Android/sdk`, the export line in the `.bashrc` file would now read as follows:

```
export  
PATH=/home/demo/java/jdk1.7.0_10/bin:/home/demo/Android/sdk/platform-  
tools:/home/demo/Android/sdk/tools:/home/demo/android-studio/bin:$PATH
```

Note also that the above command adds the `android-studio/bin` directory to the PATH variable. This will enable the `studio.sh` script to be executed regardless of the current directory within a terminal window.

### 2.8.5 Mac OS X

A number of techniques may be employed to modify the `$PATH` environment variable on Mac OS X. Arguably the cleanest method is to add a new file in the `/etc/paths.d` directory containing the paths to be added to `$PATH`. Assuming an Android SDK installation location of `/Users/demo/Library/Android/sdk`, the path may be configured by creating a new file named `android-sdk` in the `/etc/paths.d` directory containing the following lines:

```
/Users/demo/Library/Android/sdk/tools  
/Users/demo/Library/Android/sdk/platform-tools
```

Note that since this is a system directory it will be necessary to use the `sudo` command when creating the file. For example:

```
sudo vi /etc/paths.d/android-sdk
```

## 2.9 Updating the Android Studio and the SDK

From time to time new versions of Android Studio and the Android SDK are released. New versions of the SDK are installed using the Android SDK Manager. Android Studio will typically notify you when an update is ready to be installed.



To manually check for Android Studio updates, click on the *Configure -> Check for Updates* menu option within the Android Studio welcome screen, or use the *Help -> Check for Update* menu option accessible from within the Android Studio main window.

## 2.10 Summary

Prior to beginning the development of Android based applications, the first step is to set up a suitable development environment. This consists of the Java Development Kit (JDK), Android SDKs, and Android Studio IDE. In this chapter, we have covered the steps necessary to install these packages on Windows, Mac OS X and Linux.



## 3. Creating an Example Android App in Android Studio

The preceding chapters of this book have covered the steps necessary to configure an environment suitable for the development of Android applications using the Android Studio IDE. Before moving on to slightly more advanced topics, now is a good time to validate that all of the required development packages are installed and functioning correctly. The best way to achieve this goal is to create an Android application and compile and run it. This chapter will cover the creation of a simple Android application project using Android Studio. Once the project has been created, a later chapter will explore the use of the Android emulator environment to perform a test run of the application.

### 3.1 Creating a New Android Project

The first step in the application development process is to create a new project within the Android Studio environment. Begin, therefore, by launching Android Studio so that the “Welcome to Android Studio” screen appears as illustrated in Figure 3-1:

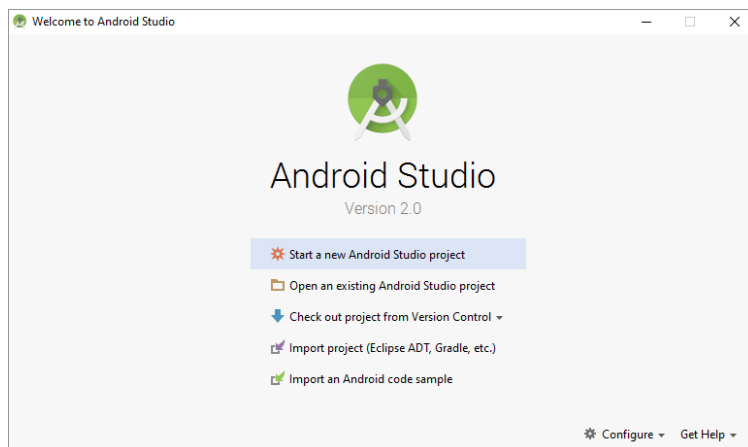


Figure 3-1

Once this window appears, Android Studio is ready for a new project to be created. To create the new project, simply click on the *Start a new Android Studio project* option to display the first screen of the *New Project* wizard as shown in Figure 3-2:

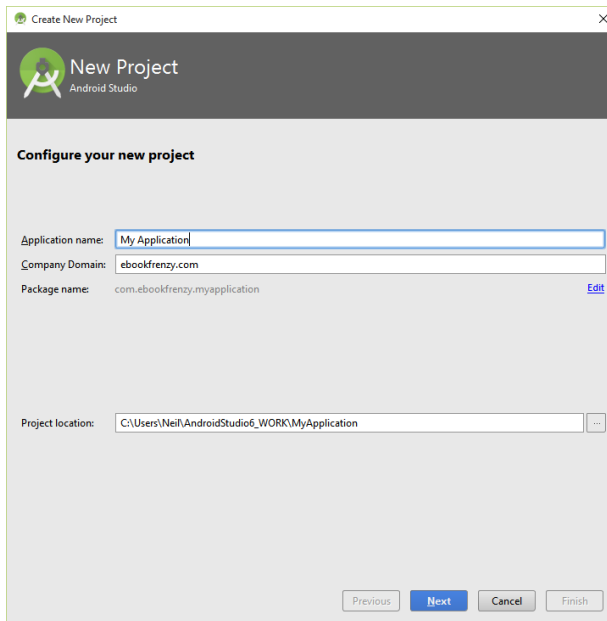


Figure 3-2

### 3.2 Defining the Project and SDK Settings

In the *New Project* window, set the *Application name* field to *AndroidSample*. The application name is the name by which the application will be referenced and identified within Android Studio and is also the name that will be used when the completed application goes on sale in the Google Play store.

The *Package Name* is used to uniquely identify the application within the Android application ecosystem. It should be based on the reversed URL of your domain name followed by the name of the application. For example, if your domain is *www.mycompany.com*, and the application has been named *AndroidSample*, then the package name might be specified as follows:

```
com.mycompany.androidsample
```

If you do not have a domain name, you may also use *ebookfrenzy.com* for the purposes of testing, though this will need to be changed before an application can be published:

```
com.ebookfrenzy.androidsample
```

The *Project location* setting will default to a location in the folder named *AndroidStudioProjects* located in your home directory and may be changed by clicking on the button to the right of the text field containing the current path setting.

Click Next to proceed. On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 8: Android 2.2 (Froyo). The reason for selecting an older SDK release is that this ensures that the finished application will be able to run on the widest possible range of Android devices. The higher the minimum SDK selection, the more the application will be restricted to newer Android devices. A useful chart (Figure 3-3) can be viewed by clicking on the *Help me choose* link. This outlines the various SDK versions and API levels available for use and the percentage of Android devices in the marketplace on which the application will run if that SDK is used as the minimum level. In general it should only be necessary to select a more recent SDK when that release contains a specific feature that is required for your application. To help in the decision process, selecting an API level from the chart will display the features that are supported at that level.

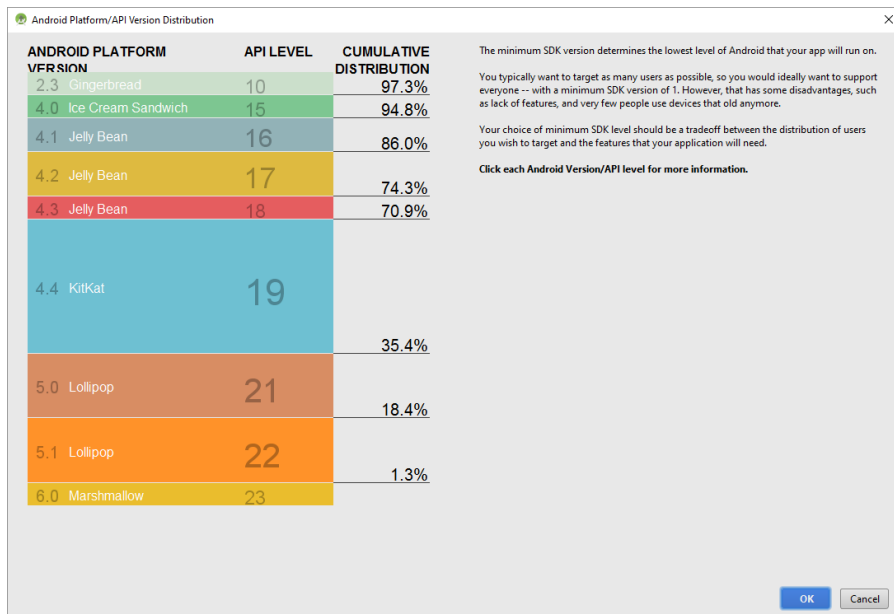


Figure 3-3

Since the project is not intended for Google TV, Android Auto or wearable devices, leave the remaining options disabled before clicking *Next*.

### 3.3 Creating an Activity

The next step is to define the type of initial activity that is to be created for the application. A range of different activity types is available when developing Android applications. The *Empty*, *Master/Detail Flow*, *Google Maps* and *Navigation Drawer* options will be covered extensively in later chapters. For the purposes of this example, however, simply select the option to create a *Basic Activity*. The Basic Activity option creates a template user interface consisting of an app bar, menu, content area and a single floating action button.

## Creating an Example Android App in Android Studio

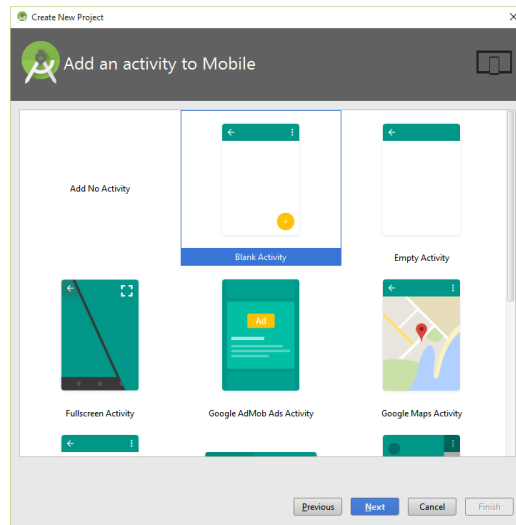


Figure 3-4

With the Basic Activity option selected, click *Next*. On the final screen (Figure 3-5) name the activity and title *AndroidSampleActivity*. The activity will consist of a single user interface screen layout which, for the purposes of this example, should be named *activity\_android\_sample* as shown in Figure 3-5 and with a menu resource named *menu\_android\_sample*:

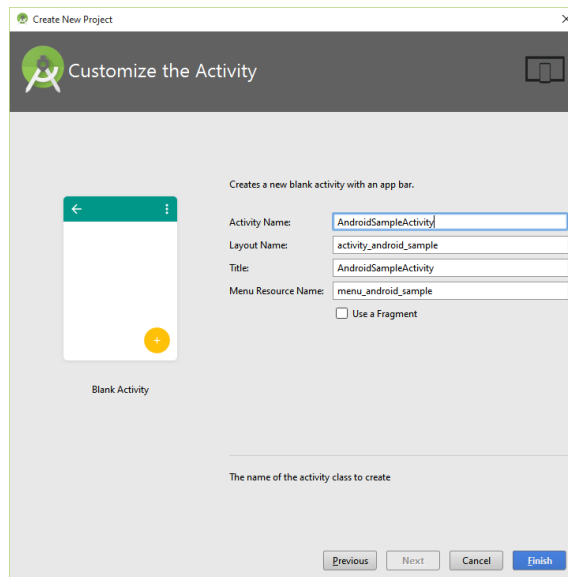


Figure 3-5

Finally, click on *Finish* to initiate the project creation process.

### 3.4 Modifying the Example Application

At this point, Android Studio has created a minimal example application project and opened the main window.

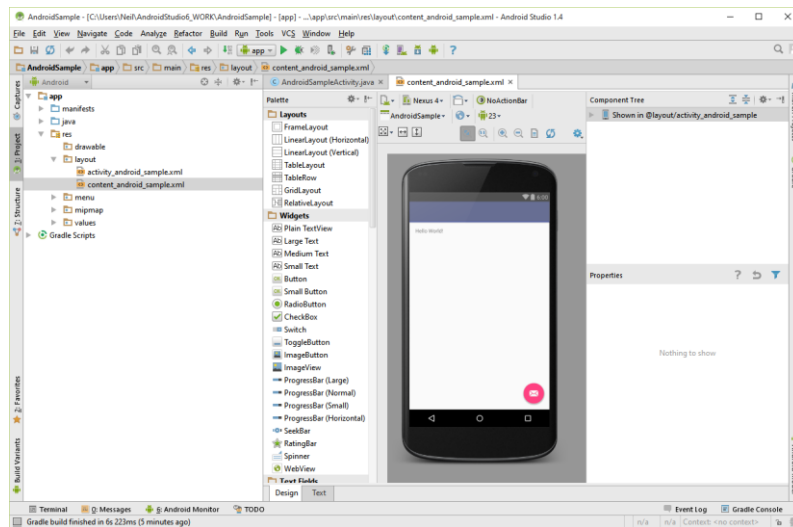


Figure 3-6

The newly created project and references to associated files are listed in the *Project* tool window located on the left hand side of the main project window. The Project tool window has a number of modes in which information can be displayed. By default, this panel will be in *Android* mode. This setting is controlled by the tabs at the top of the panel as highlighted in Figure 3-6. If the panel is not currently in Android mode, click on this tab to switch to Android mode:

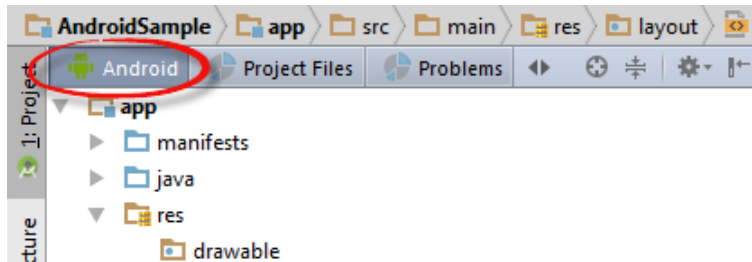


Figure 3-7

The example project created for us when we selected the option to create an activity consists of a user interface containing a label that will read “Hello World!” when the application is executed.

The next step in this tutorial is to modify the user interface of our application so that it displays a larger text view object with a different message to the one provided for us by Android Studio.

## Creating an Example Android App in Android Studio

The user interface design for our activity is stored in a file named *activity\_android\_sample.xml* which, in turn, is located under *app -> res -> layout* in the project file hierarchy. This layout file includes the app bar (also known as an action bar) that appears across the top of the device screen (marked A in Figure 3-8) and the floating action button (the email button marked B). In addition to these items, the *activity\_android\_sample.xml* layout file contains a reference to a second file containing the content layout (marked C):

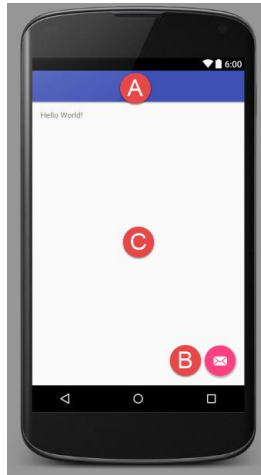


Figure 3-8

By default, the content layout is contained within a file named *content\_android\_studio.xml* and it is within this file that changes to the layout of the activity are made. Using the Project tool window, locate this file as illustrated in Figure 3-9:

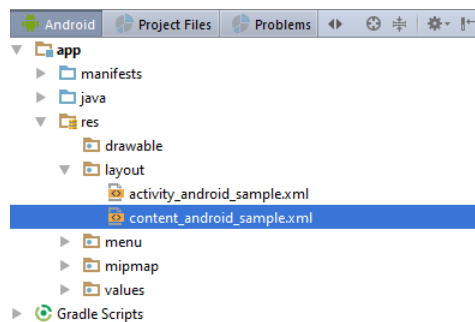


Figure 3-9

Once located, double click on the file to load it into the user interface Designer tool which will appear in the center panel of the Android Studio main window:



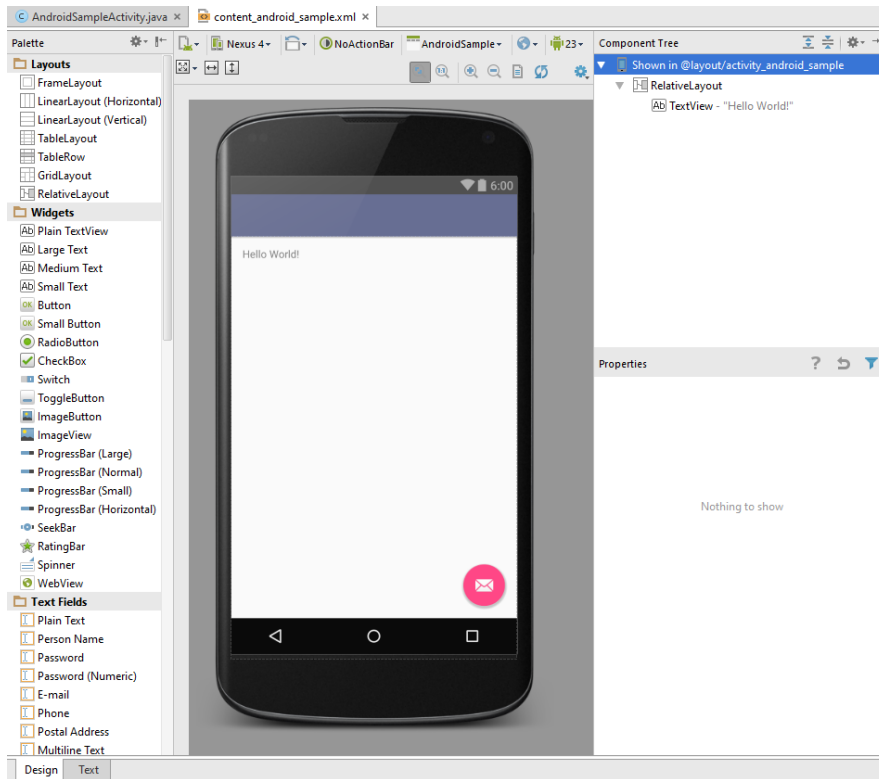



Figure 3-10

In the toolbar across the top of the Designer window is a menu currently set to *Nexus 4* which is reflected in the visual representation of the device within the Designer panel. A wide range of other device options are available for selection by clicking on this menu.

To change the orientation of the device representation between landscape and portrait simply use the drop down menu immediately to the right of the device selection menu showing the  icon.

As can be seen in the device screen, the content layout already includes a label that displays a “Hello World!” message. Running down the left hand side of the panel is a palette containing different categories of user interface components that may be used to construct a user interface, such as buttons, labels and text fields. It should be noted, however, that not all user interface components are obviously visible to the user. One such category consists of *layouts*. Android supports a variety of layouts that provide different levels of control over how visual user interface components are positioned and managed on the screen. Though it is difficult to tell from looking at the visual representation of the user interface, the current design has been created using a *RelativeLayout*. This can be confirmed by reviewing the information in the *Component Tree* panel which, by default, is located in the upper right hand corner of the Designer panel and is shown in Figure 3-11:

## Creating an Example Android App in Android Studio

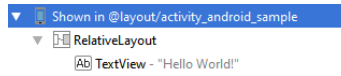


Figure 3-11

As we can see from the component tree hierarchy, the user interface layout is embedded in the *activity\_android\_sample.xml* layout file and consists of a *RelativeLayout* parent with a single child in the form of a *TextView* object.

The first step in modifying the application is to delete the *TextView* component from the design. Begin by clicking on the *TextView* object within the user interface view so that it appears with a blue border around it. Once selected, press the Delete key on the keyboard to remove the object from the layout.

In the Palette panel, locate the *Widgets* category. Click and drag the *Large Text* object and drop it in the center of the user interface design when the green marker lines appear to indicate the center of the display:

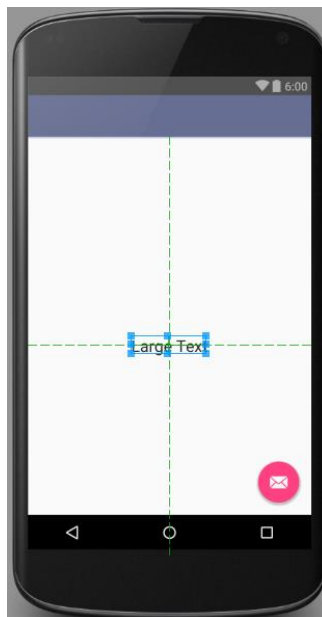


Figure 3-12

The Android Studio Designer tool also provides an alternative to dragging and dropping components from the palette on to the design layout. Components may also be added by selecting the required object from the palette and then simply clicking on the layout at the location where the component is to be placed.

The next step is to change the text that is currently displayed by the TextView component. Double click on the object in the design layout to display the text and id editing panel as illustrated in Figure 3-13. Within the panel, change the text property from “Large Text” to “Welcome to Android Studio”.

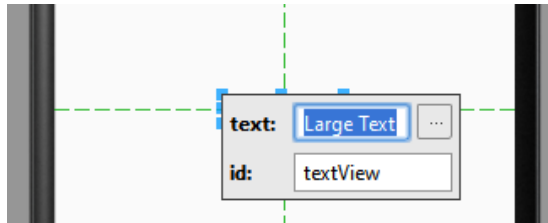


Figure 3-13

At this point it is important to explain the light bulb that has now appeared next to the TextView object in the layout. This indicates a possible problem and provides some recommended solutions. Clicking on the icon in this instance informs us that the problem is as follows:

```
[I18N] Hardcoded string "Welcome to Android Studio", should use
@string resource
```

This I18N message is informing us that a potential issue exists with regard to the future internationalization of the project (“I18N” comes from the fact that the word “internationalization” begins with an “I”, ends with an “N” and has 18 letters in between). The warning is reminding us that when developing Android applications, attributes and values such as text strings should be stored in the form of *resources* wherever possible. Doing so enables changes to the appearance of the application to be made by modifying resource files instead of changing the application source code. This can be especially valuable when translating a user interface to a different spoken language. If all of the text in a user interface is contained in a single resource file, for example, that file can be given to a translator who will then perform the translation work and return the translated file for inclusion in the application. This enables multiple languages to be targeted without the necessity for any source code changes to be made. In this instance, we are going to create a new resource named *welcomestring* and assign to it the string “Welcome to Android Studio”.

Click on the arrow to the right of the warning message to display the menu of possible solutions (Figure 3-14).

## Creating an Example Android App in Android Studio



Figure 3-14

From the menu, select the *Extract string resource* option to display the *Extract Resource* dialog. In this dialog, enter *welcomestring* into the *Resource name:* field before clicking on *OK*. The string is now stored as a resource in the *app -> res -> values -> strings.xml* file. If the layout displays the name of the string resource instead of the “Welcome to Android Studio” text, click on the refresh button located in the toolbar as highlighted in Figure 3-15:

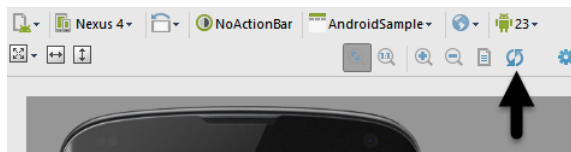


Figure 3-15

### 3.5 Reviewing the Layout and Resource Files

Before moving on to the next chapter, we are going to look at some of the internal aspects of user interface design and resource handling. In the previous section, we made some changes to the user interface by modifying the *content\_android\_sample.xml* file using the UI Designer tool. In fact, all that the Designer was doing was providing a user-friendly way to edit the underlying XML content of the file. In practice, there is no reason why you cannot modify the XML directly in order to make user interface changes and, in some instances, this may actually be quicker than using the Designer tool. At the bottom of the Designer panel are two tabs labeled *Design* and *Text* respectively. To switch to the XML view simply select the *Text* tab as shown in Figure 3-16:

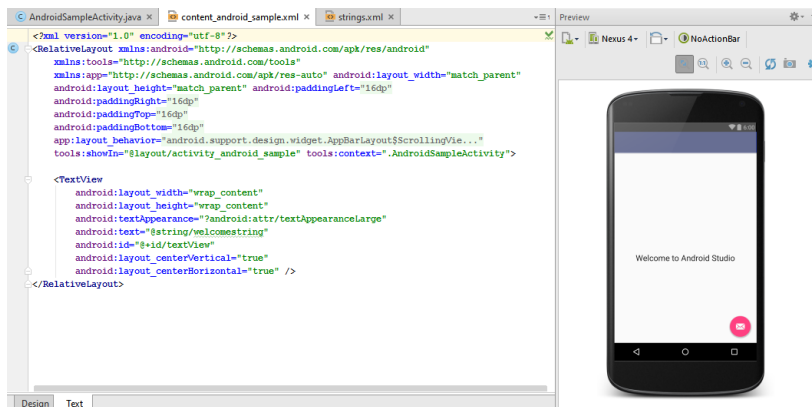


Figure 3-16

As can be seen from the structure of the XML file, the user interface consists of the `RelativeLayout` component, which in turn, is the parent of the `TextView` object. We can also see that the `text` property of the `TextView` is set to our `welcomestring` resource. Although varying in complexity and content, all user interface layouts are structured in this hierarchical, XML based way.

One of the more powerful features of Android Studio can be found to the right hand side of the XML editing panel. If the panel is not visible, display it by selecting the *Preview* button located along the right hand edge of the Android Studio window. This is the Preview panel and shows the current visual state of the layout. As changes are made to the XML layout, these will be reflected in the preview panel. To see this in action, modify the XML layout to change the background color of the `RelativeLayout` to a shade of red as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.ebookfrenzy.androidsample.AndroidSampleActivity"
    tools:showIn="@layout/activity_android_sample"
    android:background="#ff2438" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/welcome_string"
        android:id="@+id/textView"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Note that the color of the preview changes in real-time to match the new setting in the XML file. Note also that a small red square appears in the left hand margin (also referred to as the *gutter*) of the XML editor next to the line containing the color setting. This is a visual cue to the fact that the color red has been set on a property. Change the color value to `#a0ff28` and note that both the small square in the margin and the preview change to green.

## Creating an Example Android App in Android Studio

Finally, use the Project view to locate the *app* -> *res* -> *values* -> *strings.xml* file and double click on it to load it into the editor. Currently the XML should read as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">AndroidSample</string>
    <string name="action_settings">Settings</string>
    <string name="welcomestring">Welcome to Android Studio</string>

</resources>
```

As a demonstration of resources in action, change the string value currently assigned to the *welcomestring* resource and then return to the Designer tool by selecting the tab for the layout file in the editor panel. Note that the layout has picked up the new resource value for the welcome string.

There is also a quick way to access the value of a resource referenced in an XML file. With the Designer tool in Text mode, click on the “@string/welcomestring” property setting so that it highlights and then press Ctrl+B on the keyboard. Android Studio will subsequently open the *strings.xml* file and take you to the line in that file where this resource is declared. Use this opportunity to revert the string resource back to the original “Welcome to Android Studio” text.

Resource strings may also be edited using the Android Studio Translations Editor. To open this editor, right-click on the *app* -> *res* -> *values* -> *strings.xml* file and select the *Open Translations Editor* menu option. This will display the Translation Editor in the main panel of the Android Studio window:

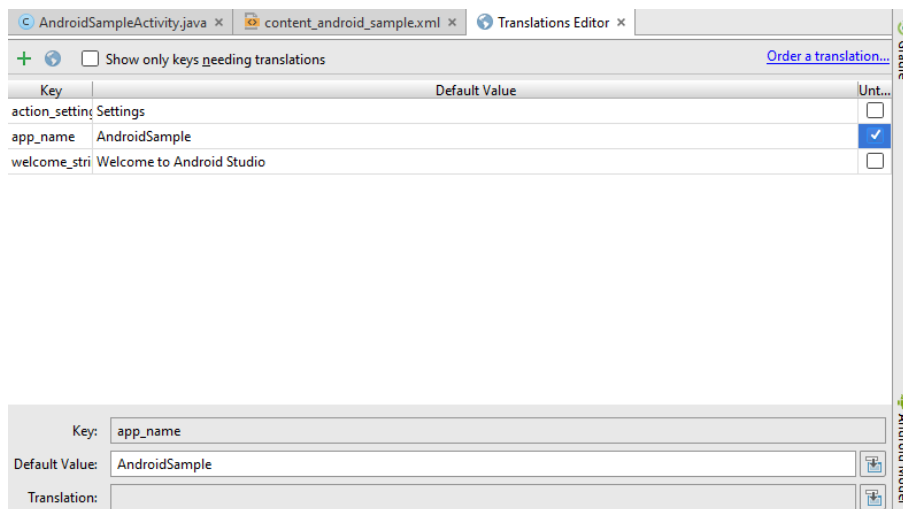


Figure 3-17

This editor allows the strings assigned to resource keys to be edited and for translations for multiple languages to be managed. The *Order a translation...* link may also be used to order a translation of the strings contained within the application to other languages. The cost of the translations will vary depending on the number of strings involved.

### 3.6 Previewing the Layout

So far in this chapter, the layout has only been previewed on a representation of the Nexus 4 device. As previously discussed, the layout can be tested for other devices by making selections from the device menu in the toolbar across the top edge of the Designer panel. Another useful option provided by this menu is *Preview All Screen Sizes* which, when selected, shows the layout in all currently configured device configurations as demonstrated in Figure 3-18.

To revert to a single preview layout, select the device menu once again, this time choosing the *Remove Previews* option.

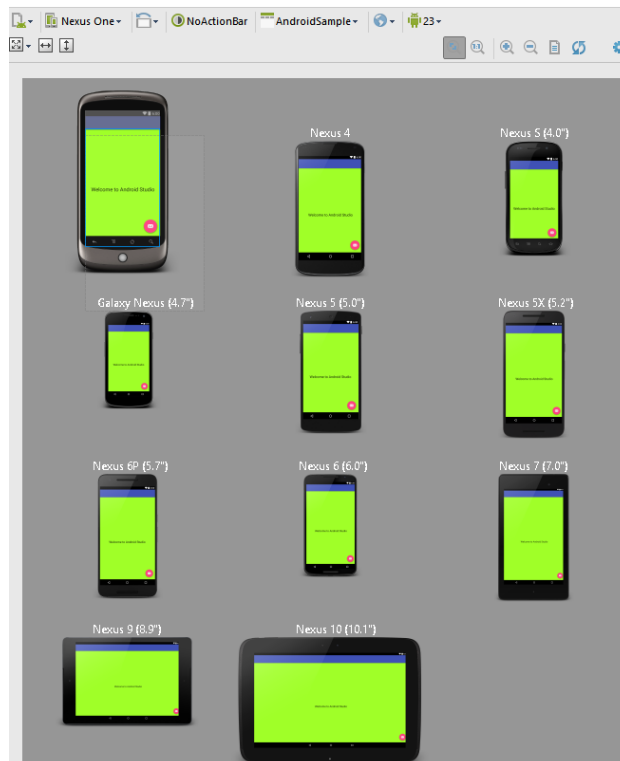


Figure 3-18

### 3.7 Summary

While not excessively complex, a number of steps are involved in setting up an Android development environment. Having performed those steps, it is worth working through a simple example to make sure the environment is correctly installed and configured. In this chapter, we have created a simple application and then used the Android Studio Designer tool to modify the user interface layout. In doing so, we explored the importance of using resources wherever possible, particularly in the case of string values, and briefly touched on the topic of layouts. Finally, we looked at the underlying XML that is used to store the user interface designs of Android applications.

While it is useful to be able to preview a layout from within the Android Studio Designer tool, there is no substitute for testing an application by compiling and running it. In a later chapter entitled *Creating an Android Virtual Device (AVD) in Android Studio*, the steps necessary to set up an emulator for testing purposes will be covered in detail. Before running the application, however, the next chapter will take a small detour to provide a guided tour of the Android Studio user interface.