Android Studio 3.0
Development Essentials
Android 8 Edition
# Table of Contents

1. Introduction .............................................................................................................................................. 1
   1.1 Downloading the Code Samples ................................................................................................. 1
   1.2 Download the eBook .................................................................................................................... 2
   1.3 Firebase Essentials Book now Available .................................................................................... 2
   1.4 Feedback ........................................................................................................................................ 2
   1.5 Errata ............................................................................................................................................. 3

2. Setting up an Android Studio Development Environment ................................................................. 5
   2.1 System Requirements .................................................................................................................. 5
   2.2 Downloading the Android Studio Package ................................................................................ 5
   2.3 Installing Android Studio .......................................................................................................... 6
      2.3.1 Installation on Windows ...................................................................................................... 6
      2.3.2 Installation on macOS .................................................................................................... 6
      2.3.3 Installation on Linux ....................................................................................................... 7
   2.4 The Android Studio Setup Wizard ............................................................................................... 7
   2.5 Installing Additional Android SDK Packages ............................................................................ 8
   2.6 Making the Android SDK Tools Command-line Accessible ...................................................... 10
      2.6.1 Windows 7 ....................................................................................................................... 11
      2.6.2 Windows 8.1 .................................................................................................................... 12
      2.6.3 Windows 10 ...................................................................................................................... 12
      2.6.4 Linux .................................................................................................................................. 13
      2.6.5 macOS ................................................................................................................................ 13
   2.7 Updating Android Studio and the SDK ....................................................................................... 13
   2.8 Summary ...................................................................................................................................... 13

3. Creating an Example Android App in Android Studio ........................................................................ 15
   3.1 Creating a New Android Project ................................................................................................ 15
   3.2 Defining the Project and SDK Settings ....................................................................................... 16
   3.3 Creating an Activity .................................................................................................................... 17
   3.4 Modifying the Example Application .......................................................................................... 19
   3.5 Reviewing the Layout and Resource Files .................................................................................. 26
   3.6 Summary ...................................................................................................................................... 29

4. A Tour of the Android Studio User Interface ...................................................................................... 31
   4.1 The Welcome Screen ................................................................................................................ 31
   4.2 The Main Window ..................................................................................................................... 32
   4.3 The Tool Windows ..................................................................................................................... 33
   4.4 Android Studio Keyboard Shortcuts .......................................................................................... 36
   4.5 Switcher and Recent Files Navigation ......................................................................................... 37
11.2.3 Service Process ................................................................. 86
11.2.4 Background Process ......................................................... 87
11.2.5 Empty Process ................................................................. 87
11.3 Inter-Process Dependencies .................................................. 87
11.4 The Activity Lifecycle ......................................................... 87
11.5 The Activity Stack ............................................................. 87
11.6 Activity States ................................................................. 88
11.7 Configuration Changes ...................................................... 89
11.8 Handling State Change ...................................................... 89
11.9 Summary ................................................................. 89

12. Handling Android Activity State Changes .................................................. 91
12.1 The Activity Class .......................................................... 91
12.2 Dynamic State vs. Persistent State ........................................ 93
12.3 The Android Activity Lifecycle Methods .............................. 94
12.4 Activity Lifetimes .......................................................... 96
12.5 Disabling Configuration Change Restarts ................................ 97
12.6 Summary ................................................................. 97

13. Android Activity State Changes by Example .................................... 99
13.1 Creating the State Change Example Project ............................ 99
13.2 Designing the User Interface ......................................... 100
13.3 Overriding the Activity Lifecycle Methods ........................... 101
13.4 Filtering the Logcat Panel ............................................ 105
13.5 Running the Application .............................................. 106
13.6 Experimenting with the Activity ..................................... 107
13.7 Summary ................................................................. 108

14. Saving and Restoring the State of an Android Activity ...................... 109
14.1 Saving Dynamic State .................................................. 109
14.2 Default Saving of User Interface State ............................... 109
14.3 The Bundle Class ....................................................... 111
14.4 Saving the State ........................................................ 111
14.5 Restoring the State ....................................................... 113
14.6 Testing the Application ................................................ 113
14.7 Summary ................................................................. 114

15. Understanding Android Views, View Groups and Layouts .................. 115
15.1 Designing for Different Android Devices ............................. 115
15.2 Views and View Groups .................................................. 115
15.3 Android Layout Managers ............................................ 116
15.4 The View Hierarchy ..................................................... 117
15.5 Creating User Interfaces ................................................ 119
22.4.6 Alignment Constraints................................................................. 190
22.4.7 Copying and Applying Constraint Sets...................................... 190
22.4.8 ConstraintLayout Chains........................................................... 190
22.4.9 Guidelines.................................................................................... 191
22.4.10 Removing Constraints............................................................... 191
22.4.11 Scaling....................................................................................... 191
22.4.12 Rotation...................................................................................... 191
22.5 Summary......................................................................................... 192

23. An Android ConstraintSet Tutorial .................................................. 193
23.1 Creating the Example Project in Android Studio............................ 193
23.2 Adding Views to an Activity............................................................ 193
23.3 Setting View Attributes................................................................. 195
23.4 Creating View IDs........................................................................... 196
23.5 Configuring the Constraint Set....................................................... 196
23.6 Adding the EditText View............................................................. 198
23.7 Converting Density Independent Pixels (dp) to Pixels (px)............. 199
23.8 Summary......................................................................................... 200

24.1 Introducing Instant Run................................................................. 201
24.2 Understanding Instant Run Swapping Levels................................. 201
24.3 Enabling and Disabling Instant Run............................................. 202
24.4 Using Instant Run............................................................................ 202
24.5 An Instant Run Tutorial................................................................. 203
24.6 Triggering an Instant Run Hot Swap............................................. 203
24.7 Triggering an Instant Run Warm Swap......................................... 204
24.8 Triggering an Instant Run Cold Swap.......................................... 204
24.9 The Run Button.............................................................................. 205
24.10 Summary....................................................................................... 205

25. An Overview and Example of Android Event Handling .................. 207
25.1 Understanding Android Events...................................................... 207
25.2 Using the android:onClick Resource............................................ 208
25.3 Event Listeners and Callback Methods......................................... 208
25.4 An Event Handling Example......................................................... 209
25.5 Designing the User Interface......................................................... 209
25.6 The Event Listener and Callback Method.................................... 210
25.7 Consuming Events........................................................................ 212
25.8 Summary....................................................................................... 214

26. Android Touch and Multi-touch Event Handling............................... 215
26.1 Intercepting Touch Events............................................................. 215
26.2 The MotionEvent Object ........................................................................................................ 216
26.3 Understanding Touch Actions ............................................................................................ 216
26.4 Handling Multiple Touches ................................................................................................. 216
26.5 An Example Multi-Touch Application ............................................................................... 217
26.6 Designing the Activity User Interface .................................................................................. 217
26.7 Implementing the Touch Event Listener ............................................................................... 218
26.8 Running the Example Application ....................................................................................... 221
26.9 Summary .............................................................................................................................. 222

27. Detecting Common Gestures using the Android Gesture Detector Class ............................ 223

27.1 Implementing Common Gesture Detection ............................................................................ 223
27.2 Creating an Example Gesture Detection Project ................................................................. 224
27.3 Implementing the Listener Class .......................................................................................... 224
27.4 Creating the GestureDetectorCompat Instance ............................................................... 227
27.5 Implementing the onTouchEvent() Method ........................................................................ 228
27.6 Testing the Application ....................................................................................................... 228
27.7 Summary .............................................................................................................................. 229

28. Implementing Custom Gesture and Pinch Recognition on Android ..................................... 231

28.1 The Android Gesture Builder Application ......................................................................... 231
28.2 The GestureOverlayView Class .......................................................................................... 231
28.3 Detecting Gestures ................................................................................................................ 231
28.4 Identifying Specific Gestures .............................................................................................. 232
28.5 Building and Running the Gesture Builder Application ...................................................... 232
28.6 Creating a Gestures File ...................................................................................................... 232
28.7 Creating the Example Project ............................................................................................. 234
28.8 Extracting the Gestures File from the SD Card .................................................................... 234
28.9 Adding the Gestures File to the Project .............................................................................. 235
28.10 Designing the User Interface ............................................................................................. 235
28.11 Loading the Gestures File .................................................................................................. 235
28.12 Registering the Event Listener .......................................................................................... 236
28.13 Implementing the onGesturePerformed Method .............................................................. 237
28.14 Testing the Application ..................................................................................................... 238
28.15 Configuring the GestureOverlayView .............................................................................. 238
28.16 Interception Gestures ........................................................................................................ 239
28.17 Detecting Pinch Gestures ................................................................................................... 239
28.18 A Pinch Gesture Example Project ..................................................................................... 240
28.19 Summary ............................................................................................................................ 242

29. An Introduction to Android Fragments ................................................................................. 243

29.1 What is a Fragment? ............................................................................................................ 243
29.2 Creating a Fragment ........................................................................................................... 243
36.9 Testing the Application ................................................................. 326
36.10 Customizing the TabLayout ........................................................... 326
36.11 Displaying Icon Tab Items ............................................................. 328
36.12 Summary .................................................................................... 329

37. Working with the RecyclerView and CardView Widgets ..................... 331
   37.1 An Overview of the RecyclerView ................................................. 331
   37.2 An Overview of the CardView .................................................... 334
   37.3 Adding the Libraries to the Project ............................................. 335
   37.4 Summary .................................................................................. 335

38. An Android RecyclerView and CardView Tutorial ............................ 337
   38.1 Creating the CardDemo Project .................................................. 337
   38.2 Removing the Floating Action Button ........................................... 337
   38.3 Adding the RecyclerView and CardView Libraries .................... 338
   38.4 Designing the CardView Layout ................................................ 338
   38.5 Adding the RecyclerView .......................................................... 340
   38.6 Creating the RecyclerView Adapter ......................................... 340
   38.7 Adding the Image Files .............................................................. 343
   38.8 Initializing the RecyclerView Component ................................... 343
   38.9 Testing the Application ............................................................. 344
   38.10 Responding to Card Selections ................................................ 345
   38.11 Summary ................................................................................. 347

39. Working with the AppBar and Collapsing Toolbar Layouts .................. 349
   39.1 The Anatomy of an AppBar ........................................................ 349
   39.2 The Example Project ................................................................ 350
   39.3 Coordinating the RecyclerView and Toolbar .............................. 350
   39.4 Introducing the Collapsing Toolbar Layout .................................. 352
   39.5 Changing the Title and Scrim Color ........................................... 356
   39.6 Summary .................................................................................. 357

40. Implementing an Android Navigation Drawer ..................................... 359
   40.1 An Overview of the Navigation Drawer ....................................... 359
   40.2 Opening and Closing the Drawer ............................................... 361
   40.3 Responding to Drawer Item Selections ....................................... 361
   40.4 Using the Navigation Drawer Activity Template .......................... 362
   40.5 Creating the Navigation Drawer Template Project ...................... 362
   40.6 The Template Layout Resource Files ........................................ 363
   40.7 The Header Coloring Resource File ......................................... 363
   40.8 The Template Menu Resource File .......................................... 363
   40.9 The Template Code .................................................................. 363
   40.10 Running the App ..................................................................... 364
41. An Android Studio Master/Detail Flow Tutorial ................................................................. 367
   41.1 The Master/Detail Flow ........................................................................................................ 367
   41.2 Creating a Master/Detail Flow Activity ............................................................................ 368
   41.3 The Anatomy of the Master/Detail Flow Template ......................................................... 370
   41.4 Modifying the Master/Detail Flow Template ................................................................. 371
   41.5 Changing the Content Model ......................................................................................... 371
   41.6 Changing the Detail Pane ............................................................................................... 373
   41.7 Modifying the WebsiteDetailFragment Class ............................................................... 374
   41.8 Modifying the WebsiteListActivity Class .................................................................. 376
   41.9 Adding Manifest Permissions ....................................................................................... 376
   41.10 Running the Application ............................................................................................. 377
   41.11 Summary ..................................................................................................................... 377

42. An Overview of Android Intents ......................................................................................... 379
   42.1 An Overview of Intents ..................................................................................................... 379
   42.2 Explicit Intents .................................................................................................................. 379
   42.3 Returning Data from an Activity ..................................................................................... 381
   42.4 Implicit Intents ................................................................................................................ 382
   42.5 Using Intent Filters ......................................................................................................... 382
   42.6 Checking Intent Availability .......................................................................................... 383
   42.7 Summary ......................................................................................................................... 384

43. Android Explicit Intents – A Worked Example ............................................................... 385
   43.1 Creating the Explicit Intent Example Application .......................................................... 385
   43.2 Designing the User Interface Layout for ActivityA ....................................................... 385
   43.3 Creating the Second Activity Class ................................................................................ 387
   43.4 Designing the User Interface Layout for ActivityB ....................................................... 387
   43.5 Reviewing the Application Manifest File ...................................................................... 388
   43.6 Creating the Intent ......................................................................................................... 389
   43.7 Extracting Intent Data ..................................................................................................... 390
   43.8 Launching ActivityB as a Sub-Activity ......................................................................... 391
   43.9 Returning Data from a Sub-Activity .............................................................................. 392
   43.10 Testing the Application ............................................................................................... 393
   43.11 Summary ..................................................................................................................... 393

44. Android Implicit Intents – A Worked Example ............................................................... 395
   44.1 Creating the Android Studio Implicit Intent Example Project ..................................... 395
   44.2 Designing the User Interface .......................................................................................... 395
   44.3 Creating the Implicit Intent ............................................................................................ 396
   44.4 Adding a Second Matching Activity ............................................................................. 397
   44.5 Adding the Web View to the UI .................................................................................... 397
48.3 Adding the Service to the Manifest File ................................................................. 429
48.4 Starting the Service .............................................................................................. 430
48.5 Testing the IntentService Example ...................................................................... 430
48.6 Using the Service Class ....................................................................................... 431
48.7 Creating the New Service .................................................................................. 431
48.8 Modifying the User Interface ............................................................................. 433
48.9 Running the Application ..................................................................................... 434
48.10 Creating an AsyncTask for Service Tasks ....................................................... 434
48.11 Summary ............................................................................................................ 436

49. Android Local Bound Services – A Worked Example ........................................ 437

49.1 Understanding Bound Services .......................................................................... 437
49.2 Bound Service Interaction Options .................................................................... 437
49.3 An Android Studio Local Bound Service Example ......................................... 438
49.4 Adding a Bound Service to the Project ............................................................. 438
49.5 Implementing the Binder .................................................................................... 439
49.6 Binding the Client to the Service ....................................................................... 441
49.7 Completing the Example .................................................................................... 443
49.8 Testing the Application ...................................................................................... 444
49.9 Summary ............................................................................................................. 445

50. Android Remote Bound Services – A Worked Example ...................................... 447

50.1 Client to Remote Service Communication ....................................................... 447
50.2 Creating the Example Application ..................................................................... 447
50.3 Designing the User Interface ............................................................................. 448
50.4 Implementing the Remote Bound Service ......................................................... 448
50.5 Configuring a Remote Service in the Manifest File ......................................... 449
50.6 Launching and Binding to the Remote Service ............................................... 450
50.7 Sending a Message to the Remote Service ....................................................... 452
50.8 Summary ............................................................................................................. 452

51. An Android 8 Notifications Tutorial .................................................................. 453

51.1 An Overview of Notifications ............................................................................ 453
51.2 Creating the NotifyDemo Project ...................................................................... 455
51.3 Designing the User Interface ............................................................................. 455
51.4 Creating the Second Activity ............................................................................. 456
51.5 Creating a Notification Channel ...................................................................... 456
51.6 Creating and Issuing a Basic Notification ......................................................... 459
51.7 Launching an Activity from a Notification ....................................................... 462
51.8 Adding Actions to a Notification ...................................................................... 463
51.9 Bundled Notifications ....................................................................................... 464
51.10 Summary ............................................................................................................. 466
62.5 Configuring the VideoView .............................................................. 564
62.6 Adding Internet Permission ........................................................... 565
62.7 Adding the MediaController to the Video View .......................... 566
62.8 Setting up the onPreparedListener ............................................. 567
62.9 Summary ....................................................................................... 568

63. Android Picture-in-Picture Mode ....................................................... 569
   63.1 Picture-in-Picture Features ......................................................... 569
   63.2 Enabling Picture-in-Picture Mode ............................................. 570
   63.3 Configuring Picture-in-Picture Parameters .............................. 571
   63.4 Entering Picture-in-Picture Mode ............................................. 571
   63.5 Detecting Picture-in-Picture Mode Changes ............................. 572
   63.6 Adding Picture-in-Picture Actions ............................................ 572
   63.7 Summary ..................................................................................... 573

64. An Android Picture-in-Picture Tutorial ............................................. 575
   64.1 Changing the Minimum SDK Setting ........................................ 575
   64.2 Adding Picture-in-Picture Support to the Manifest ................. 575
   64.3 Adding a Picture-in-Picture Button .......................................... 576
   64.4 Entering Picture-in-Picture Mode ............................................ 576
   64.5 Detecting Picture-in-Picture Mode Changes ............................ 578
   64.6 Adding a Broadcast Receiver .................................................. 579
   64.7 Adding the PIP Action .............................................................. 580
   64.8 Testing the Picture-in-Picture Action ....................................... 583
   64.9 Summary ..................................................................................... 584

65. Video Recording and Image Capture on Android using Camera Intents .................................................. 585
   65.1 Checking for Camera Support ................................................... 585
   65.2 Calling the Video Capture Intent .............................................. 585
   65.3 Calling the Image Capture Intent .............................................. 587
   65.4 Creating an Android Studio Video Recording Project ............... 587
   65.5 Designing the User Interface Layout ....................................... 588
   65.6 Checking for the Camera .......................................................... 588
   65.7 Launching the Video Capture Intent ........................................ 589
   65.8 Handling the Intent Return ...................................................... 590
   65.9 Testing the Application ........................................................... 591
   65.10 Summary .................................................................................. 591

66. Making Runtime Permission Requests in Android ........................... 593
   66.1 Understanding Normal and Dangerous Permissions ................ 593
   66.2 Creating the Permissions Example Project .............................. 595
   66.3 Checking for a Permission ....................................................... 595
   66.4 Requesting Permission at Runtime .......................................... 597
66.5 Providing a Rationale for the Permission Request ........................................... 599
66.6 Testing the Permissions App ............................................................................. 600
66.7 Summary ......................................................................................................... 601

67. Android Audio Recording and Playback using MediaPlayer and MediaRecorder ............................................................................. 603

67.1 Playing Audio ................................................................................................. 603
67.2 Recording Audio and Video using the MediaRecorder Class ......................... 604
67.3 About the Example Project ............................................................................ 605
67.4 Creating the AudioApp Project ..................................................................... 605
67.5 Designing the User Interface ......................................................................... 606
67.6 Checking for Microphone Availability ............................................................ 606
67.7 Performing the Activity Initialization ............................................................... 607
67.8 Implementing the recordAudio() Method ....................................................... 609
67.9 Implementing the stopAudio() Method ........................................................... 610
67.10 Implementing the playAudio() method ......................................................... 610
67.11 Configuring and Requesting Permissions .................................................... 611
67.12 Testing the Application ................................................................................ 614
67.13 Summary ....................................................................................................... 614

68. Working with the Google Maps Android API in Android Studio ..................... 617

68.1 The Elements of the Google Maps Android API ........................................... 617
68.2 Creating the Google Maps Project ................................................................. 618
68.3 Obtaining Your Developer Signature .............................................................. 618
68.4 Testing the Application .................................................................................. 619
68.5 Understanding Geocoding and Reverse Geocoding ...................................... 620
68.6 Adding a Map to an Application .................................................................... 622
68.7 Requesting Current Location Permission ...................................................... 622
68.8 Displaying the User’s Current Location .......................................................... 624
68.9 Changing the Map Type ................................................................................ 625
68.10 Displaying Map Controls to the User ............................................................. 626
68.11 Handling Map Gesture Interaction ................................................................ 627
  68.11.1 Map Zooming Gestures .......................................................................... 627
  68.11.2 Map Scrolling/Panning Gestures ............................................................. 627
  68.11.3 Map Tilt Gestures .................................................................................. 628
  68.11.4 Map Rotation Gestures .......................................................................... 628
68.12 Creating Map Markers .................................................................................. 628
68.13 Controlling the Map Camera ......................................................................... 629
68.14 Summary ....................................................................................................... 631

69. Printing with the Android Printing Framework .............................................. 633

69.1 The Android Printing Architecture ................................................................. 633
69.2 The Print Service Plugins .............................................................................. 633
69.3 Google Cloud Print .................................................................................................................. 634
69.4 Printing to Google Drive ......................................................................................................... 635
69.5 Save as PDF ............................................................................................................................... 635
69.6 Printing from Android Devices ............................................................................................... 635
69.7 Options for Building Print Support into Android Apps ......................................................... 637
    69.7.1 Image Printing .................................................................................................................. 637
    69.7.2 Creating and Printing HTML Content ............................................................................. 638
    69.7.3 Printing a Web Page ....................................................................................................... 639
    69.7.4 Printing a Custom Document ....................................................................................... 640
69.8 Summary .................................................................................................................................. 640

70. An Android HTML and Web Content Printing Example ....................................................... 641
    70.1 Creating the HTML Printing Example Application ........................................................... 641
    70.2 Printing Dynamic HTML Content ..................................................................................... 641
    70.3 Creating the Web Page Printing Example .......................................................................... 644
    70.4 Removing the Floating Action Button ................................................................................ 645
    70.5 Designing the User Interface Layout .................................................................................. 645
    70.6 Loading the Web Page into the WebView ......................................................................... 647
    70.7 Adding the Print Menu Option ........................................................................................... 648
    70.8 Summary ................................................................................................................................ 650

    71.1 An Overview of Android Custom Document Printing .................................................... 651
        71.1.1 Custom Print Adapters ............................................................................................... 651
    71.2 Preparing the Custom Document Printing Project ............................................................ 652
    71.3 Creating the Custom Print Adapter .................................................................................... 653
    71.4 Implementing the onLayout() Callback Method ............................................................... 654
    71.5 Implementing the onWrite() Callback Method ................................................................. 657
    71.6 Checking a Page is in Range .............................................................................................. 660
    71.7 Drawing the Content on the Page Canvas ......................................................................... 661
    71.8 Starting the Print Job ......................................................................................................... 664
    71.9 Testing the Application ....................................................................................................... 665
    71.10 Summary ........................................................................................................................... 666

72. An Introduction to Android App Links ...................................................................................... 667
    72.1 An Overview of Android App Links .................................................................................... 667
    72.2 App Link Intent Filters ........................................................................................................ 667
    72.3 Handling App Link Intents ................................................................................................ 668
    72.4 Associating the App with a Website ................................................................................... 669
    72.5 Summary ............................................................................................................................. 670

73. An Android Studio App Links Tutorial .................................................................................... 671
    73.1 About the Example App ...................................................................................................... 671
1. Introduction

Fully updated for Android Studio 3.0 and Android 8, the goal of this book is to teach the skills necessary to develop Android based applications using the Android Studio Integrated Development Environment (IDE), the Android 8 Software Development Kit (SDK) and the Java programming language.

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 Layout Editor 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, and submitting apps to the Google Play Developer Console.

Other key features of Android Studio 3 and Android 8 are also covered in detail including the Layout Editor, the ConstraintLayout and ConstraintSet classes, constraint chains and barriers, direct reply notifications and multi-window support.

Chapters also cover advanced features of Android Studio such as App Links, Instant Apps, the Android Studio Profiler and Gradle build configuration.

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/androidstudio30/index.php

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

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 and we will provide you with a download link for the book in PDF format.

1.3 Firebase Essentials Book Now Available


The Firebase Essentials book covers the key features of Android app development using Firebase including integration with Android Studio, User Authentication (including email, Twitter, Facebook and phone number sign-in), Realtime Database, Cloud Storage, Firebase Cloud Messaging (both upstream and downstream), Dynamic Links, Invites, App Indexing, Test Lab, Remote Configuration, Cloud Functions, Analytics and Performance Monitoring.

Find out more at https://goo.gl/5F381e.

1.4 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.
1.5 **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/androidstudio30.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. They are there to help you and will work to resolve any problems you may encounter.
Chapter 2

2. Setting up an Android Studio Development Environment

Before 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 Android Studio Integrated Development Environment (IDE) which also includes the Android Software Development Kit (SDK) and OpenJDK Java development environment.

This chapter will cover the steps necessary to install the requisite components for Android application development on Windows, macOS 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)
- macOS 10.10 or later (Intel based systems only)
- Linux systems with version 2.19 or later of GNU C Library (glibc)
- Minimum of 3GB of RAM (8GB is preferred)
- Approximately 4GB of available disk space
- 1280 x 800 minimum screen resolution

2.2 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 3.0.

At the time of writing, both Android Studio 3.0 and the Android 8 SDK are available in preview versions only. The location for downloading the Android Studio package will depend on whether or not the software is still in preview. Begin by checking the primary download page for Android Studio which can be found at the following URL:


If this page provides instructions for downloading Android Studio 3.0, perform the download from this page. If, on the other hand, the page provides access to Android Studio 2.3, you will need to download the latest preview edition of Android Studio 3.0 from the following web page:

Setting up an Android Studio Development Environment

From the appropriate page, select the package for your platform and operating system.

2.3 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.

2.3.1 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.

2.3.2 Installation on macOS

Android Studio for macOS 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 2-1:

![Figure 2-1](image)
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.

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

### 2.3.3 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:

```bash
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:

```bash
./studio.sh
```

When running on a 64-bit Linux system, it will be necessary to install some 32-bit support libraries before Android Studio will run. On Ubuntu these libraries can be installed using the following command:

```bash
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1 libbz2-1.0:i386
```

On RedHat and Fedora based 64-bit systems, use the following command:

```bash
sudo yum install zlib.i686 ncurses-libs.i686 bzip2-libs.i686
```

### 2.4 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 2-2 though this dialog does not appear on all platforms:
If the wizard appears, click on the Next button, choose the Standard installation option and click on Next once again.

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:

2.5 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 2-4:

![Android SDK screen](image)

**Figure 2-4**

Immediately after installing Android Studio for the first time it is likely that only the latest released 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 2-5:

![Android SDK update](image)

**Figure 2-5**

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.

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 2-6:
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 Emulator
- Android SDK Platform-tools
- Android SDK Tools
- Google Play Services
- Instant Apps Development SDK
- Intel x86 Emulator Accelerator (HAXM installer)
- ConstraintLayout for Android
- Solver for ConstraintLayout
- Android Support Repository
- Google Repository
- Google USB Driver (Windows only)

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 *Apply* button again.

### 2.6 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/tools/bin
<path_to_android_sdk_installation>/sdk/platform-tools
```

The location of the SDK on your system can be identified by launching the SDK Manager and referring to the `Android SDK Location:` field located at the top of the settings panel as highlighted in Figure 2-7:

![Figure 2-7](image)

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

### 2.6.1 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;
C:\Users\demo\AppData\Local\Android\sdk\tools\bin
```

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
```
Setting up an Android Studio Development Environment

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

Similarly, check the tools path setting by attempting to launch the AVD Manager command line tool:

```bash
avdmanager
```

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.
```

### 2.6.2 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:

```bash
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:

```bash
adb
```

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

Similarly, check the tools path setting by attempting to run the AVD Manager command line tool:

```
avdmanager
```

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.
```

### 2.6.3 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.
2.6.4 Linux

On Linux, this configuration 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). Assuming that the Android SDK bundle package was installed into /home/demo/Android/sdk, the export line in the .bashrc file would read as follows:

```bash
export PATH=/home/demo/Android/sdk/platform-tools:/home/demo/Android/sdk/tools:/home/demo/Android/sdk/tools/bin:/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.

2.6.5 macOS

A number of techniques may be employed to modify the $PATH environment variable on macOS. 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/tools/bin
/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
```

2.7 Updating 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 Update 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.

2.8 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, macOS 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:
Creating an Example Android App in Android Studio

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:

![Create Android Project](image)

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. Although this can be set to any string that uniquely identifies your app, it is traditionally 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 can enter any other string into the Company Domain field, or you may use example.com for the purposes of testing, though this will need to be changed before an application can be published:

```
com.example.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 14: Android 4.0 (IceCreamSandwich). 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.

Since the project is not intended for Google TV, Android Auto or wearable devices, leave the remaining options disabled before clicking *Next*. Instant Apps will not be covered until later in this book so make sure that the *Include Android Instant App support* option is disabled.

### 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

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`. Finally, enter *My Android App* into the title field as shown in Figure 3-5:

Since the `AndroidSampleActivity` is essentially the top level activity for the project and has no parent activity, there is no need to specify an activity for the Hierarchical parent (in other words `AndroidSampleActivity` does not need an “Up” button to return to another activity).

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.

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 menu at the top of the panel as highlighted in Figure 3-7. If the panel is not currently in Android mode, use the menu to switch mode:
Creating an Example Android App in Android Studio

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.

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](image)

By default, the content layout is contained within a file named content_android_sample.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:
Once located, double-click on the file to load it into the user interface Layout Editor tool which will appear in the center panel of the Android Studio main window:

In the toolbar across the top of the Layout Editor window is a menu (currently set to Nexus 4 in the above figure) which is reflected in the visual representation of the device within the Layout Editor panel. A wide range of other device options are available for selection by clicking on this menu.
Creating an Example Android App in Android Studio

To change the orientation of the device representation between landscape and portrait simply use the drop down menu immediately to the left 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 ConstraintLayout. This can be confirmed by reviewing the information in the Component Tree panel which, by default, is located in the lower left-hand corner of the Layout Editor panel and is shown in Figure 3-11:

![Component Tree]

As we can see from the component tree hierarchy, the user interface layout consists of a ConstraintLayout parent with a single child in the form of a TextView object.

Before proceeding, check that the Layout Editor’s Autoconnect mode is enabled. This means that as components are added to the layout, the Layout Editor will automatically add constraints to make sure the components are correctly positioned for different screen sizes and device orientations (a topic that will be covered in much greater detail in future chapters). The Autoconnect button appears in the Layout Editor toolbar and is represented by a magnet icon. When disabled the magnet appears with a diagonal line through it (Figure 3-12). If necessary, re-enable Autoconnect mode by clicking on this button.

![Figure 3-12]

The next 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.

The Palette panel consists of two columns with the left-hand column containing a list of view component categories. The right-hand column lists the components contained within the currently selected category. In Figure 3-13, for example, the Button view is currently selected within the Widgets category:
Click and drag the *Button* object (either from the Widgets list, or the preview area) and drop it in the center of the user interface design when the marker lines appear indicating the center of the display:
Creating an Example Android App in Android Studio

The next step is to change the text that is currently displayed by the Button component. The panel located to the right of the design area is the Attributes panel. This panel displays the attributes assigned to the currently selected component in the layout. Within this panel, locate the text property and change the current value from “Button” to “Demo” as shown in Figure 3-15:

![Attributes Panel with Demo text](image)

**Figure 3-15**

A useful shortcut to changing the text property of a component is to double-click on it in the layout. This will automatically locate the attribute in the attributes panel and select it ready for editing.

The second text property with a wrench next to it allows a text property to be set which only appears within the Layout Editor tool but is not shown at runtime. This is useful for testing the way in which a visual component and the layout will behave with different settings without having to run the app repeatedly.

At this point it is important to explain the warning button located in the top right-hand corner of the Layout Editor tool as indicated in Figure 3-16. Obviously, this is indicating potential problems with the layout. For details on any problems, click on the button:

![Warning Button](image)

**Figure 3-16**

When clicked, a panel (Figure 3-17) will appear describing the nature of the problems and offering some possible corrective measures:
Currently, the only warning listed reads as follows:

```
Hardcoded string “Demo”, 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 `demostring` and assign to it the string “Demo”.

Click on the Fix button in the Issue Explanation panel to display the Extract Resource panel (Figure 3-18). Within this panel, change the resource name field to `demostring` and leave the resource value set to Demo before clicking on the OK button.
It is also worth noting that the string could also have been assigned to a resource when it was entered into the Attributes panel. This involves clicking on the button displaying three dots to the right of the property field in the Attributes panel and selecting the Add new resource -> New String Value... menu option from the resulting Resources dialog. In practice, however, it is often quicker to simply set values directly into the Attributes panel fields for any widgets in the layout, then work sequentially through the list in the warnings dialog to extract any necessary resources when the layout is complete.

### 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 Layout Editor tool. In fact, all that the Layout Editor 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 Layout Editor tool. At the bottom of the Layout Editor panel are two tabs labeled Design and Text respectively. To switch to the XML view simply select the Text tab as shown in Figure 3-19:
As can be seen from the structure of the XML file, the user interface consists of the ConstraintLayout component, which in turn, is the parent of the Button object. We can also see that the text property of the Button is set to our demostring 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. The layout may also be modified visually from within the Preview panel with the changes appearing in the XML listing. To see this in action, modify the XML layout to change the background color of the ConstraintLayout to a shade of red as follows:

```xml
<android.support.constraint.ConstraintLayout
    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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.ebookfrenzy.myapplication.AndroidSampleActivity"
    tools:showIn="@layout/activity_android_sample"
    android:background="#ff2438" />
```
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.

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
<resources>
  <string name="app_name">AndroidSample</string>
  <string name="action_settings">Settings</string>
  <string name="demostring">Demo</string>
</resources>
```

As a demonstration of resources in action, change the string value currently assigned to the `demostring` resource to “Hello” and then return to the Layout Editor 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 string.

There is also a quick way to access the value of a resource referenced in an XML file. With the Layout Editor tool in Text mode, click on the “@string/demostring” 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 “Demo” 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 Editor menu option. This will display the Translation Editor in the main panel of the Android Studio window:

![Transations Editor](image)

**Figure 3-20**

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 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 Layout Editor 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 Layout Editor 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.
4. A Tour of the Android Studio User Interface

While it is tempting to plunge into running the example application created in the previous chapter, doing so involves using aspects of the Android Studio user interface which are best described in advance.

Android Studio is a powerful and feature rich development environment that is, to a large extent, intuitive to use. That being said, taking the time now to gain familiarity with the layout and organization of the Android Studio user interface will considerably shorten the learning curve in later chapters of the book. With this in mind, this chapter will provide an initial overview of the various areas and components that make up the Android Studio environment.

4.1 The Welcome Screen

The welcome screen (Figure 4-1) is displayed any time that Android Studio is running with no projects currently open (open projects can be closed at any time by selecting the File -> Close Project menu option). If Android Studio was previously exited while a project was still open, the tool will by-pass the welcome screen next time it is launched, automatically opening the previously active project.

In addition to a list of recent projects, the Quick Start menu provides a range of options for performing tasks such as opening, creating and importing projects along with access to projects currently under version control.
In addition, the *Configure* menu at the bottom of the window provides access to the SDK Manager along with a vast array of settings and configuration options. A review of these options will quickly reveal that there is almost no aspect of Android Studio that cannot be configured and tailored to your specific needs.

The Configure menu also includes an option to check if updates to Android Studio are available for download.

### 4.2 The Main Window

When a new project is created, or an existing one opened, the Android Studio *main window* will appear. When multiple projects are open simultaneously, each will be assigned its own main window. The precise configuration of the window will vary depending on which tools and panels were displayed the last time the project was open, but will typically resemble that of Figure 4-2.

![Figure 4-2](image)

The various elements of the main window can be summarized as follows:

**A – Menu Bar** – Contains a range of menus for performing tasks within the Android Studio environment.

**B – Toolbar** – A selection of shortcuts to frequently performed actions. The toolbar buttons provide quicker access to a select group of menu bar actions. The toolbar can be customized by right-clicking on the bar and selecting the *Customize Menus and Toolbars...* menu option.

**C – Navigation Bar** – The navigation bar provides a convenient way to move around the files and folders that make up the project. Clicking on an element in the navigation bar will drop down a menu listing the subfolders and files at that location ready for selection. This provides an alternative to the Project tool window.

**D – Editor Window** – The editor window displays the content of the file on which the developer is currently working. What gets displayed in this location, however, is subject to context. When editing code, for example, the code editor will appear. When working on a user interface layout file, on the other hand, the user interface
Layout Editor tool will appear. When multiple files are open, each file is represented by a tab located along the top edge of the editor as shown in Figure 4-3.

**E – Status Bar** – The status bar displays informational messages about the project and the activities of Android Studio together with the tools menu button located in the far left corner. Hovering over items in the status bar will provide a description of that field. Many fields are interactive, allowing the user to click to perform tasks or obtain more detailed status information.

**F – Project Tool Window** – The project tool window provides a hierarchical overview of the project file structure allowing navigation to specific files and folders to be performed. The toolbar can be used to display the project in a number of different ways. The default setting is the *Android* view which is the mode primarily used in the remainder of this book.

The project tool window is just one of a number of tool windows available within the Android Studio environment.

### 4.3 The Tool Windows

In addition to the project view tool window, Android Studio also includes a number of other windows which, when enabled, are displayed along the bottom and sides of the main window. The tool window quick access menu can be accessed by hovering the mouse pointer over the button located in the far left-hand corner of the status bar (Figure 4-4) without clicking the mouse button.
Selecting an item from the quick access menu will cause the corresponding tool window to appear within the main window.

Alternatively, a set of tool window bars can be displayed by clicking on the quick access menu icon in the status bar. These bars appear along the left, right and bottom edges of the main window (as indicated by the arrows in Figure 4-5) and contain buttons for showing and hiding each of the tool windows. When the tool window bars are displayed, a second click on the button in the status bar will hide them.

![Figure 4-5](image)

Clicking on a button will display the corresponding tool window while a second click will hide the window. Buttons prefixed with a number (for example 1: Project) indicate that the tool window may also be displayed by pressing the Alt key on the keyboard (or the Command key for macOS) together with the corresponding number.

The location of a button in a tool window bar indicates the side of the window against which the window will appear when displayed. These positions can be changed by clicking and dragging the buttons to different locations in other window tool bars.

Each tool window has its own toolbar along the top edge. The buttons within these toolbars vary from one tool to the next, though all tool windows contain a settings option, represented by the cog icon, which allows various aspects of the window to be changed. Figure 4-6 shows the settings menu for the project view tool window. Options are available, for example, to undock a window and to allow it to float outside of the boundaries of the Android Studio main window and to move and resize the tool panel.
All of the windows also include a far right button on the toolbar providing an additional way to hide the tool window from view. A search of the items within a tool window can be performed simply by giving that window focus by clicking in it and then typing the search term (for example the name of a file in the Project tool window). A search box will appear in the window’s tool bar and items matching the search highlighted.

Android Studio offers a wide range of tool windows, the most commonly used of which are as follows:

**Project** – The project view provides an overview of the file structure that makes up the project allowing for quick navigation between files. Generally, double-clicking on a file in the project view will cause that file to be loaded into the appropriate editing tool.

**Structure** – The structure tool provides a high level view of the structure of the source file currently displayed in the editor. This information includes a list of items such as classes, methods and variables in the file. Selecting an item from the structure list will take you to that location in the source file in the editor window.

**Captures** – The captures tool window provides access to performance data files that have been generated by the monitoring tools contained within Android Studio.

**Favorites** – A variety of project items can be added to the favorites list. Right-clicking on a file in the project view, for example, provides access to an *Add to Favorites* menu option. Similarly, a method in a source file can be added as a favorite by right-clicking on it in the Structure tool window. Anything added to a Favorites list can be accessed through this Favorites tool window.

**Build Variants** – The build variants tool window provides a quick way to configure different build targets for the current application project (for example different builds for debugging and release versions of the application, or multiple builds to target different device categories).

**TODO** – As the name suggests, this tool provides a place to review items that have yet to be completed on the project. Android Studio compiles this list by scanning the source files that make up the project to look for comments that match specified TODO patterns. These patterns can be reviewed and changed by selecting...
the File -> Settings... menu option (Android Studio -> Preferences... on macOS) and navigating to the TODO page listed under Editor.

**Messages** – The messages tool window records output from the Gradle build system (Gradle is the underlying system used by Android Studio for building the various parts of projects into runnable applications) and can be useful for identifying the causes of build problems when compiling application projects.

**Logcat** – The Logcat tool window provides access to the monitoring log output from a running application in addition to options for taking screenshots and videos of the application and stopping and restarting a process.

**Terminal** – Provides access to a terminal window on the system on which Android Studio is running. On Windows systems this is the Command Prompt interface, while on Linux and macOS systems this takes the form of a Terminal prompt.

**Run** – The run tool window becomes available when an application is currently running and provides a view of the results of the run together with options to stop or restart a running process. If an application is failing to install and run on a device or emulator, this window will typically provide diagnostic information relating to the problem.

**Event Log** – The event log window displays messages relating to events and activities performed within Android Studio. The successful build of a project, for example, or the fact that an application is now running will be reported within this tool window.

**Gradle Console** – The Gradle console is used to display all output from the Gradle system as projects are built from within Android Studio. This will include information about the success or otherwise of the build process together with details of any errors or warnings.

**Gradle** – The Gradle tool window provides a view onto the Gradle tasks that make up the project build configuration. The window lists the tasks that are involved in compiling the various elements of the project into an executable application. Right-click on a top level Gradle task and select the Open Gradle Config menu option to load the Gradle build file for the current project into the editor. Gradle will be covered in greater detail later in this book.

**Android Profiler** – The Android Profiler tool window provides realtime monitoring and analysis tools for identifying performance issues within running apps, including CPU, memory and network usage.

**Device File Explorer** – The Device File Explorer tool window provides direct access to the filesystem of the currently connected Android device or emulator allowing the filesystem to be browsed and files copied to the local filesystem.

### 4.4 Android Studio Keyboard Shortcuts

Android Studio includes an abundance of keyboard shortcuts designed to save time when performing common tasks. A full keyboard shortcut keymap listing can be viewed and printed from within the Android Studio project window by selecting the Help -> Keymap Reference menu option.
4.5 Switcher and Recent Files Navigation

Another useful mechanism for navigating within the Android Studio main window involves the use of the **Switcher**. Accessed via the **Ctrl-Tab** keyboard shortcut, the switcher appears as a panel listing both the tool windows and currently open files (Figure 4-7).

![Figure 4-7](image1.png)

Once displayed, the switcher will remain visible for as long as the Ctrl key remains depressed. Repeatedly tapping the Tab key while holding down the Ctrl key will cycle through the various selection options, while releasing the Ctrl key causes the currently highlighted item to be selected and displayed within the main window.

In addition to the switcher, navigation to recently opened files is provided by the Recent Files panel (Figure 4-8). This can be accessed using the **Ctrl-E** keyboard shortcut (**Cmd-E** on macOS). Once displayed, either the mouse pointer can be used to select an option or, alternatively, the keyboard arrow keys used to scroll through the file name and tool window options. Pressing the Enter key will select the currently highlighted item.

![Figure 4-8](image2.png)
4.6 Changing the Android Studio Theme

The overall theme of the Android Studio environment may be changed either from the welcome screen using the Configure -> Settings option, or via the File -> Settings... menu option (Android Studio -> Preferences... on macOS) of the main window.

Once the settings dialog is displayed, select the Appearance option in the left-hand panel and then change the setting of the Theme menu before clicking on the Apply button. The themes available will depend on the platform but usually include options such as IntelliJ, Windows, Default and Darcula. Figure 4-9 shows an example of the main window with the Darcula theme selected:

![Example of main window with Darcula theme selected](image)

Figure 4-9

4.7 Summary

The primary elements of the Android Studio environment consist of the welcome screen and main window. Each open project is assigned its own main window which, in turn, consists of a menu bar, toolbar, editing and design area, status bar and a collection of tool windows. Tool windows appear on the sides and bottom edges of the main window and can be accessed either using the quick access menu located in the status bar, or via the optional tool window bars.

There are very few actions within Android Studio which cannot be triggered via a keyboard shortcut. A keymap of default keyboard shortcuts can be accessed at any time from within the Android Studio main window.
5. Creating an Android Virtual Device (AVD) in Android Studio

In the course of developing Android apps in Android Studio it will be necessary to compile and run an application multiple times. An Android application may be tested by installing and running it either on a physical device or in an Android Virtual Device (AVD) emulator environment. Before an AVD can be used, it must first be created and configured to match the specifications of a particular device model. The goal of this chapter, therefore, is to work through the steps involved in creating such a virtual device using the Nexus 5X phone as a reference example.

5.1 About Android Virtual Devices

AVDs are essentially emulators that allow Android applications to be tested without the necessity to install the application on a physical Android based device. An AVD may be configured to emulate a variety of hardware features including options such as screen size, memory capacity and the presence or otherwise of features such as a camera, GPS navigation support or an accelerometer. As part of the standard Android Studio installation, a number of emulator templates are installed allowing AVDs to be configured for a range of different devices. Additional templates may be loaded or custom configurations created to match any physical Android device by specifying properties such as processor type, memory capacity and the size and pixel density of the screen. Check the online developer documentation for your device to find out if emulator definitions are available for download and installation into the AVD environment.

When launched, an AVD will appear as a window containing an emulated Android device environment. Figure 5-1, for example, shows an AVD session configured to emulate the Google Nexus 5X model.

New AVDs are created and managed using the Android Virtual Device Manager, which may be used either in command-line mode or with a more user-friendly graphical user interface.
5.2 Creating a New AVD

In order to test the behavior of an application in the absence of a physical device, it will be necessary to create an AVD for a specific Android device configuration.

To create a new AVD, the first step is to launch the AVD Manager. This can be achieved from within the Android Studio environment by selecting the Tools -> Android -> AVD Manager menu option from within the main window.

Once launched, the tool will appear as outlined in Figure 5-2 if existing AVD instances have been created:
To add an additional AVD, begin by clicking on the Create Virtual Device button in order to invoke the Virtual Device Configuration dialog:

Within the dialog, perform the following steps to create a Nexus 5X compatible emulator:

1. From the Category panel, select the Phone option to display the list of available Android tablet AVD templates.
2. Select the Nexus 5X device option and click Next.
3. On the System Image screen, select the latest version of Android (at time of writing this is Oreo, API level 26, Android 8.0 with Google Play) for the x86 ABI. Note that if the system image has not yet been installed a Download link will be provided next to the Release Name. Click this link to download and install the system image before selecting it. If the image you need is not listed, click on the x86 images and Other images tabs to view alternative lists.
4. Click Next to proceed and enter a descriptive name (for example Nexus 5X API 26) into the name field or simply accept the default name.
5. Click Finish to create the AVD.

With the AVD created, the AVD Manager may now be closed. If future modifications to the AVD are necessary, simply re-open the AVD Manager, select the AVD from the list and click on the pencil icon in the Actions column of the device row in the AVD Manager.

5.3 Starting the Emulator

To perform a test run of the newly created AVD emulator, simply select the emulator from the AVD Manager and click on the launch button (the green triangle in the Actions column). The emulator will appear in a new window and begin the startup process. The amount of time it takes for the emulator to start will depend on the configuration of both the AVD and the system on which it is running. In the event that the startup time on your system is considerable, do not hesitate to leave the emulator running. The system will detect that it
Creating an Android Virtual Device (AVD) in Android Studio

is already running and attach to it when applications are launched, thereby saving considerable amounts of startup time.

The emulator probably defaulted to appearing in portrait orientation. It is useful to be aware that this and other default options can be changed. Within the AVD Manager, select the new Nexus 5X entry and click on the pencil icon in the Actions column of the device row. In the configuration screen locate the Startup and orientation section and change the orientation setting. Exit and restart the emulator session to see this change take effect. More details on the emulator are covered in the next chapter (Using and Configuring the Android Studio AVD Emulator).

To save time in the next section of this chapter, leave the emulator running before proceeding.

5.4 Running the Application in the AVD

With an AVD emulator configured, the example AndroidSample application created in the earlier chapter now can be compiled and run. With the AndroidSample project loaded into Android Studio, simply click on the run button represented by a green triangle located in the Android Studio toolbar as shown in Figure 5-4 below, select the Run -> Run ‘app’ menu option or use the Ctrl-R keyboard shortcut:

![Figure 5-4](image)

By default, Android Studio will respond to the run request by displaying the Select Deployment Target dialog. This provides the option to execute the application on an AVD instance that is already running, or to launch a new AVD session specifically for this application. Figure 5-5 lists the previously created Nexus 5X AVD as a running device as a result of the steps performed in the preceding section. With this device selected in the dialog, click on OK to install and run the application on the emulator.

![Figure 5-5](image)
Once the application is installed and running, the user interface for the AndroidSampleActivity class will appear within the emulator:

![AndroidSample](image)

**Figure 5-6**

In the event that the activity does not automatically launch, check to see if the launch icon has appeared among the apps on the emulator. If it has, simply click on it to launch the application. Once the run process begins, the Run and Logcat tool windows will become available. The Run tool window will display diagnostic information as the application package is installed and launched. Figure 5-7 shows the Run tool window output from a successful application launch:

![Run tool output](image)

**Figure 5-7**

If problems are encountered during the launch process, the Run tool window will provide information that will hopefully help to isolate the cause of the problem.

Assuming that the application loads into the emulator and runs as expected, we have safely verified that the Android development environment is correctly installed and configured.

### 5.5 Run/Debug Configurations

A particular project can be configured such that a specific device or emulator is used automatically each time it is run from within Android Studio. This avoids the necessity to make a selection from the device chooser.
Creating an Android Virtual Device (AVD) in Android Studio

each time the application is executed. To review and modify the Run/Debug configuration, click on the button to the left of the run button in the Android Studio toolbar and select the *Edit Configurations*... option from the resulting menu:

![Edit Configurations](image)

**Figure 5-8**

In the *Run/Debug Configurations* dialog, the application may be configured to always use a preferred emulator by selecting *Emulator* from the *Target* menu located in the *Deployment Target Options* section and selecting the emulator from the drop down menu. Figure 5-9, for example, shows the AndroidSample application configured to run by default on the previously created Nexus 5X emulator:

![Run/Debug Configurations](image)

**Figure 5-9**

Be sure to switch the Target menu setting back to "Open Select Deployment Target Dialog" mode before moving on to the next chapter of the book.
5.6 Stopping a Running Application

To stop a running application, simply click on the stop button located in the main toolbar as shown in Figure 5-10:

![Stop 'app' (F2)](image)

An app may also be terminated using the Logcat tool window. Begin by displaying the Logcat tool window either using the window bar button, or via the quick access menu (invoked by moving the mouse pointer over the button in the left-hand corner of the status bar as shown in Figure 5-11).

![Figure 5-11](image)

Once the Logcat tool window appears, select the androidsample app menu highlighted in Figure 5-12 below:

![Figure 5-12](image)

With the process selected, stop it by clicking on the red Terminate Application button in the toolbar to the left of the process list indicated by the arrow in the above figure.
Creating an Android Virtual Device (AVD) in Android Studio

An alternative to using the Android tool window is to open the Android Device Monitor. This can be launched via the Tools -> Android -> Android Device Monitor menu option. Once launched, the process may be selected from the list (Figure 5-13) and terminated by clicking on the red Stop button located in the toolbar above the list.

![Android Device Monitor](image)

**Figure 5-13**

### 5.7 AVD Command-line Creation

As previously discussed, in addition to the graphical user interface it is also possible to create a new AVD directly from the command-line. This is achieved using the `avdmanager` tool in conjunction with some command-line options. Once initiated, the tool will prompt for additional information before creating the new AVD.

Assuming that the system has been configured such that the Android SDK `tools` directory is included in the PATH environment variable, a list of available targets for the new AVD may be obtained by issuing the following command in a terminal or command window:

```
avdmanager list targets
```

The resulting output from the above command will contain a list of Android SDK versions that are available on the system. For example:

```
Available Android targets:
-------------------------
id: 1 or "android-25"
   Name: Android API 25
```
Creating an Android Virtual Device (AVD) in Android Studio

The avdmanager tool also allows new AVD instances to be created from the command line. For example, to create a new AVD named Nexus9 using the target ID for the Android API level 26 device using the x86 ABI, the following command may be used:

```
avdmanager create avd -n Nexus9 -k "system-images;android-26;googleApis;x86"
```

The android tool will create the new AVD to the specifications required for a basic Android 8 device, also providing the option to create a custom configuration to match the specification of a specific device if required. Once a new AVD has been created from the command line, it may not show up in the Android Device Manager tool until the Refresh button is clicked.

In addition to the creation of new AVDs, a number of other tasks may be performed from the command line. For example, a list of currently available AVDs may be obtained using the list avd command line arguments:

```
avdmanager list avd
```

Available Android Virtual Devices:
Name: Nexus_5X_API_26
  Device: Nexus 5X (Google)
  Path: /Users/neilsmyth/.android/avd/Nexus_5X_API_26.avd
  Target: Google Play (Google Inc.)
    Based on: Android 8.0 (Oreo) Tag/ABI:
      googleApis_playstore/x86
  Skin: nexus_5x
  Sdcard: 100M

Similarly, to delete an existing AVD, simply use the delete option as follows:

```
avdmanager delete avd -n <avd name>
```

### 5.8 Android Virtual Device Configuration Files

By default, the files associated with an AVD are stored in the .android/avd sub-directory of the user’s home directory, the structure of which is as follows (where <avd name> is replaced by the name assigned to the AVD):

```
<avd name>.avd/config.ini
```
Creating an Android Virtual Device (AVD) in Android Studio

<avd name>.avd/userdata.img
<avd name>.ini

The config.ini file contains the device configuration settings such as display dimensions and memory specified during the AVD creation process. These settings may be changed directly within the configuration file and will be adopted by the AVD when it is next invoked.

The <avd name>.ini file contains a reference to the target Android SDK and the path to the AVD files. Note that a change to the image.sysdir value in the config.ini file will also need to be reflected in the target value of this file.

5.9 Moving and Renaming an Android Virtual Device

The current name or the location of the AVD files may be altered from the command line using the avdmanager tool's move avd argument. For example, to rename an AVD named Nexus9 to Nexus9B, the following command may be executed:

```
avdmanager move avd -n Nexus9 -r Nexus9B
```

To physically relocate the files associated with the AVD, the following command syntax should be used:

```
avdmanager move avd -n <avd name> -p <path to new location>
```

For example, to move an AVD from its current file system location to /tmp/Nexus9Test:

```
avdmanager move avd -n Nexus9 -p /tmp/Nexus9Test
```

Note that the destination directory must not already exist prior to executing the command to move an AVD.

5.10 Summary

A typical application development process follows a cycle of coding, compiling and running in a test environment. Android applications may be tested on either a physical Android device or using an Android Virtual Device (AVD) emulator. AVDs are created and managed using the Android AVD Manager tool which may be used either as a command line tool or using a graphical user interface. When creating an AVD to simulate a specific Android device model it is important that the virtual device be configured with a hardware specification that matches that of the physical device.