# Android Studio Development Essentials

Android Studio Development Essentials – First Edition

ISBN-13: 978-0-9860273-7-6

Rev 1.0

eBookFrenzy

# Table of Contents

Chapter 1

# 1. Introduction

The goal of this book is to teach the skills necessary to develop Android based applications using the Android Studio Integrated Development Environment (IDE) and the Android 4.4 Software Development Kit (SDK).

Beginning with the basics, this book provides an outline of the steps necessary to set up an Android development and testing environment. An overview of Android Studio is included covering areas such as tool windows, the code editor and the Designer tool. An introduction to the architecture of Android is followed by an in-depth look at the design of Android applications and user interfaces using the Android Studio environment. More advanced topics such as database management, content providers and intents are also covered, as are touch screen handling, gesture recognition, camera access and the playback and recording of both video and audio. This edition of the book also covers features introduced with Android 4.4 including printing, transitions and cloud-based file storage.

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

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

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

## 1.1 Downloading the Code Samples

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

*http://www.ebookfrenzy.com/print/androidstudio/index.php*

## 1.2 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*.

## 1.3 Errata

Whilst 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/androidstudio.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 Java Development Kit (JDK) and the Android Studio Integrated Development Environment (IDE) which also includes the Android Software Development Kit (SDK).

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

## 2.1 System Requirements

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

- Windows XP (32-bit)
- Windows Vista (32-bit or 64-bit)
- Windows 7 (32-bit or 64-bit)
- Windows 8 / Windows 8.1
- Mac OS X 10.5.8 or later (Intel based systems only)
- Linux systems with version 2.7 or later of GNU C Library (glibc)

## 2.2 Installing the Java Development Kit (JDK)

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

Android development requires the installation of the Standard Edition of the Java Platform Development Kit version 6 or later. Java is provided in both development (JDK) and runtime (JRE) packages. For the purposes of Android development, the JDK must be installed.

### 2.2.1 Windows JDK Installation

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

*http://www.oracle.com/technetwork/java/javase/downloads/index.html*

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

### 2.2.2 Mac OS X JDK Installation

The Java SE 6 environment or a more recent version should already be installed on the latest Mac OS X versions. To confirm the version that is installed, open a Terminal window and enter the following command:

```
java -version
```

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

```
java version "1.6.0_65"
Java(TM) SE Runtime Environment (build 1.6.0_65-b14-462-11M4609)
Java HotSpot(TM) 64-Bit Server VM (build 20.65-b04-462, mixed mode)
```

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

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

The exact steps that need to be taken to install Java vary from one release of Mac OS X to the next so check the Apple documentation for your particular Mac OS X version.

At the time of writing the latest release of Mac OS X is 10.9 (Mavericks). To install Java on this release of Mac OS X, open a Safari browser window and navigate to the following URL:

*http://support.apple.com/kb/DL1572*

This should display the Java for OS X 2013-005 web page. Click on the *Download* button to download the Java package to your system. Open the downloaded disk image (.dmg file) and double click on the *JavaForOSX.pkg* package file (Figure 2-1) contained within:

Figure 2-1

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

```
java -version
```

## 2.3 Linux JDK Installation

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

```
sudo apt-get install ia32-libs
```

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

*http://www.oracle.com/technetwork/java/javase/downloads/index.html*

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

On Red Hat based Linux systems, download the .rpm JDK file from the Oracle web site and perform the installation using the *rpm* command in a terminal window. Assuming, for example, that the downloaded JDK file was named *jdk-7u45-linux-x64.rpm*, the commands to perform the installation would read as follows:

```
su
rpm –ihv jdk-7u45-linux-x64.rpm
```

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

1. Create the directory into which the JDK is to be installed (for the purposes of this example we will assume */home/demo/java*).

2. Download the appropriate tar.gz package from the Oracle web site into the directory.

3. Execute the following command (where *<jdk-file>* is replaced by the name of the downloaded JDK file):

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

4. Remove the downloaded tar.gz file.

5. Add the path to the *bin* directory of the JDK installation to your $PATH variable. For example, assuming that the JDK ultimately installed into */home/demo/java/jdk1.7.0_45* the following would need to be added to your $PATH environment variable:

```
/home/demo/java/jdk1.7.0_45/bin
```

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

```
export PATH=/home/demo/java/jdk1.7.0_45/bin:$PATH
```

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

## 2.4 Downloading the Android Studio Package

Most of the work involved in developing applications for Android will be performed using the Android Studio environment. Android Studio may be downloaded from the following web page:

http://developer.android.com/sdk/installing/studio.html

From this page, either click on the download button if it lists the correct platform (for example on a Windows based web browser the button will read “Download Android Studio for Windows”), or select the “Download for Other Platforms” option to manually select the appropriate package for your platform and operating system. On the subsequent screen, accept the terms and conditions to initiate the download.

## 2.5 **Installing Android Studio**

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

### 2.5.1 Installation on Windows

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

Once the Android Studio setup wizard appears, work through the various screens to configure the installation to meet your requirements in terms of the file system location into which Android Studio should be installed and whether or not it should be made available to other users of the system. 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 a sub-folder of the user's home directory named *android-studio*. 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. On Windows 8, the executable can be pinned to the task bar for easy access by navigating to the *android-studio\bin* directory, right-clicking on the executable and selecting the *Pin to Taskbar* menu option. Note that the executable is provided in 32-bit (*studio*) and 64-bit (*studio64*) executable versions. If you are running a 32-bit system be sure to use the *studio* executable.

### 2.5.2 Installation on Mac OS X

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

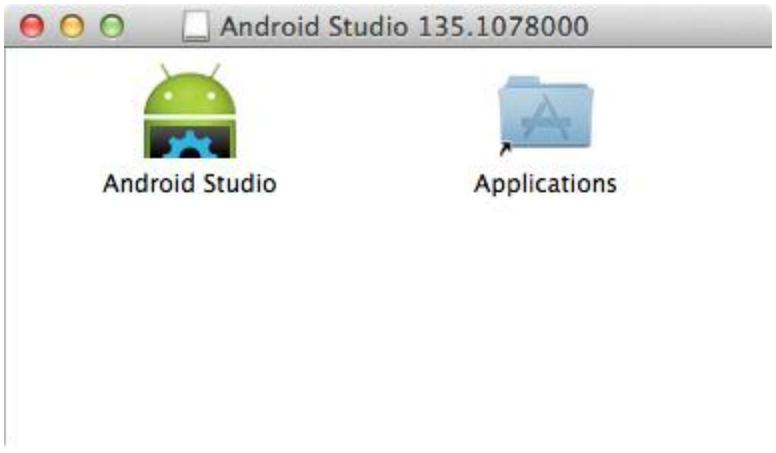


Figure 2-2

To install the package, simply drag the Android Studio icon and drop it onto the Applications icon. 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.

### 2.5.3 Installation on Linux

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

```
tar xvf /<path to package>/android-studio-bundle-<version>.tgz
```

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

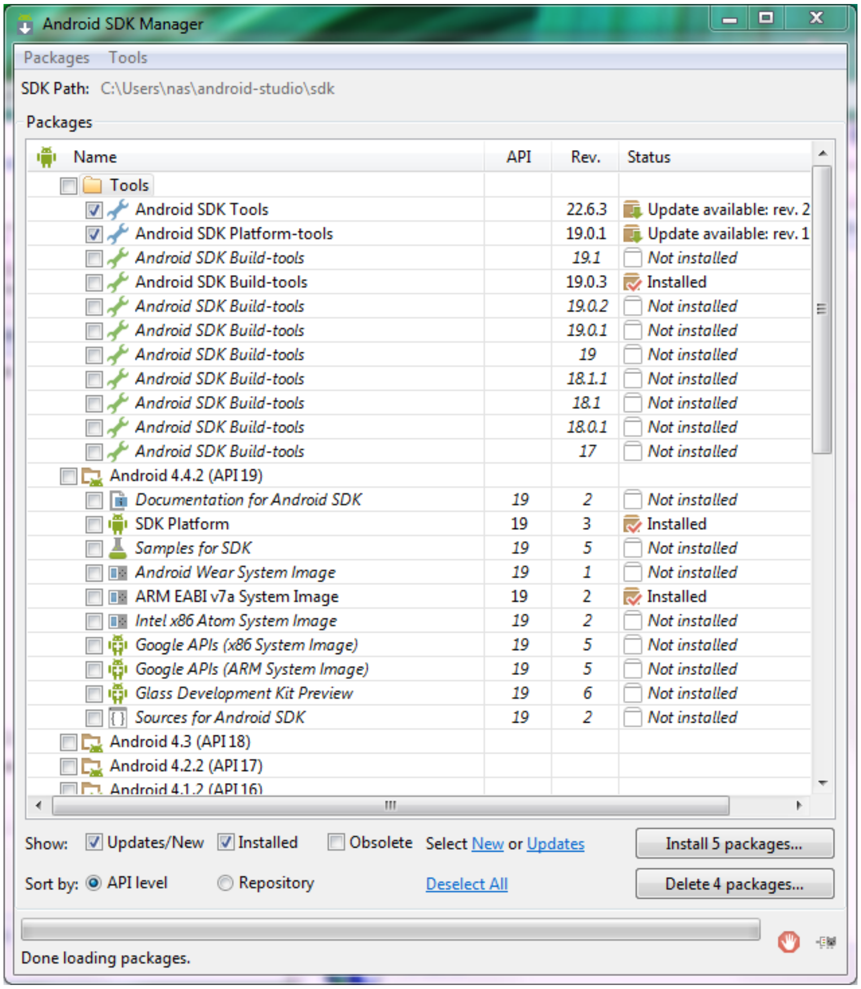
To launch Android Studio, open a terminal window, change directory to the *android-studio/bin* sub-directory and execute the following command:

```
./studio.sh
```

## 2.6 Installing the Latest 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 packages.

This task can be performed using the *Android SDK Manager*, 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 SDK Manager tool will appear as illustrated in Figure 2-3:



Android SDK Manager

Packages Tools

SDK Path: C:\Users\nas\android-studio\sdk

Packages

| Name | API | Rev. | Status |
|---|---|---|---|
| Tools | | | |
| [✓] Android SDK Tools | | 22.6.3 | Update available: rev. 2 |
| [✓] Android SDK Platform-tools | | 19.0.1 | Update available: rev. 1 |
| *Android SDK Build-tools* | | *19.1* | *Not installed* |
| Android SDK Build-tools | | 19.0.3 | Installed |
| *Android SDK Build-tools* | | *19.0.2* | *Not installed* |
| *Android SDK Build-tools* | | *19.0.1* | *Not installed* |
| *Android SDK Build-tools* | | *19* | *Not installed* |
| *Android SDK Build-tools* | | *18.1.1* | *Not installed* |
| *Android SDK Build-tools* | | *18.1* | *Not installed* |
| *Android SDK Build-tools* | | *18.0.1* | *Not installed* |
| *Android SDK Build-tools* | | *17* | *Not installed* |
| Android 4.4.2 (API 19) | | | |
| *Documentation for Android SDK* | *19* | *2* | *Not installed* |
| SDK Platform | 19 | 3 | Installed |
| *Samples for SDK* | *19* | *5* | *Not installed* |
| *Android Wear System Image* | *19* | *1* | *Not installed* |
| ARM EABI v7a System Image | 19 | 2 | Installed |
| *Intel x86 Atom System Image* | *19* | *2* | *Not installed* |
| *Google APIs (x86 System Image)* | *19* | *5* | *Not installed* |
| *Google APIs (ARM System Image)* | *19* | *5* | *Not installed* |
| *Glass Development Kit Preview* | *19* | *6* | *Not installed* |
| *Sources for Android SDK* | *19* | *2* | *Not installed* |
| Android 4.3 (API 18) | | | |
| Android 4.2.2 (API 17) | | | |
| Android 4.1.2 (API 16) | | | |

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates Install 5 packages...

Sort by: [✓ API level] Repository Deselect All Delete 4 packages...

Done loading packages.

Figure 2-3

Begin by checking that the *SDK Path:* setting at the top of the SDK Manager window matches the location into which the Android Studio was installed. Within the Android SDK Manager, make sure that the check boxes next to the following packages are listed as *Installed* in the Status column:

- Tools > Android SDK Tools
- Tools > Android SDK Platform-tools
- SDK Platform (most recent version)> SDK Platform
- SDK Platform (most recent version) > ARM EABI v7a System Image
  Note that the SDK Platform is now available in standard, L and W variants. For the purposes of this book it is not necessary to download the L and W versions of the SDK platform packages.
- Extras -> Android Support Repository
- Extras > Android Support Library
- Extras -> Google Repository
- Extras -> Google USB Driver

In the event that any of the above packages are listed as *Not Installed*, simply select the checkboxes next to those packages and click on the *Install packages* button to initiate the installation process. In the resulting dialog, accept the license agreements before clicking on the *Install* button. The SDK Manager will then begin to download and install the designated packages. As the installation proceeds, a progress bar will appear at the bottom of the manager window indicating the status of the installation.

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

## 2.7 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_studio_installation>* represents the file system location into which the ADT bundle was installed):

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

The steps to achieve this are operating system dependent:

### 2.7.1 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 Android Studio was installed into */Users/demo/android-studio*, the following would be appended to the end of the current Path value:

```
;C:\Users\demo\android-studio\sdk\platform-tools;C:\Users\demo\android-
studio\sdk\tools
```

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

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

```
echo %Path%
```

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

```
adb
```

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

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

```
android
```

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

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

### 2.7.2 Windows 8.1

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

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

Within the Command Prompt window, enter:

```
echo %Path%
```

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

```
adb
```

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

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

```
android
```

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

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

### 2.7.3 Linux

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

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

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

### 2.7.4 Mac OS X

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

```
/Applications/Android Studio.app/sdk/tools
/Applications/Android Studio.app/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
```

## 2.8 Updating the Android Studio and the SDK

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

To manually check for Android Studio updates, click on the *Check for updates now* link located at the bottom of the Android Studio welcome screen, or use the *Help -> Check for Update…* menu option accessible from within the Android Studio main window.

## 2.9 Summary

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

Chapter 3

# 3. Creating an Example Android App in Android Studio

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

## 3.1 Creating a New Android Project

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

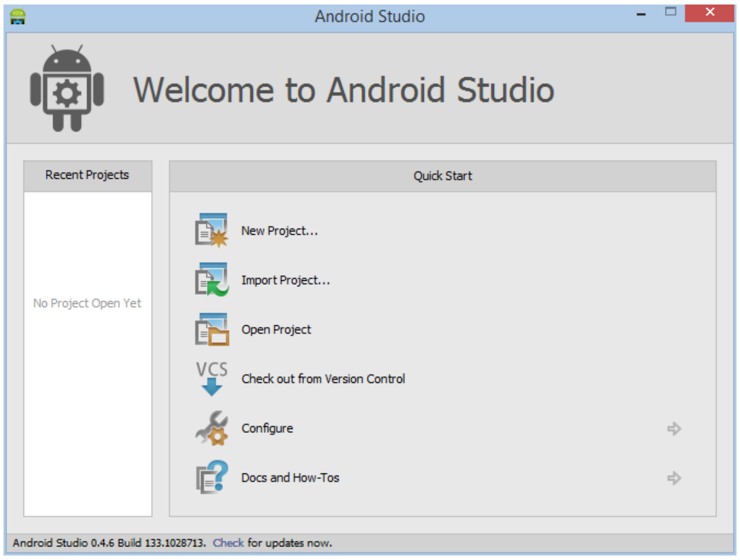
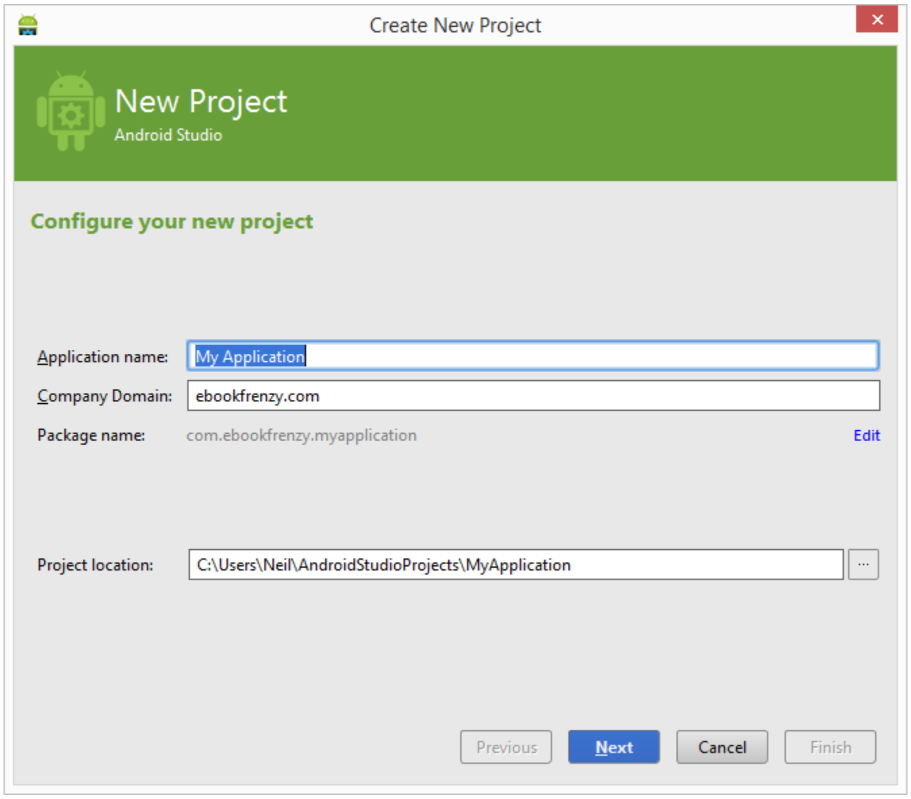


Figure 3-1

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



Application name: My Application
Company Domain: ebookfrenzy.com
Package name: com.ebookfrenzy.myapplication
Project location: C:\Users\Neil\AndroidStudioProjects\MyApplication

Figure 3-2

## 3.2 Defining the Project and SDK Settings

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

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

```
com.mycompany.androidsample
```

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

```
com.ebookfrenzy.androidsample
```

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

Click Next to proceed. On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 8: Android 2.2 (Froyo). Since the project is not intended for Google TV, Google Glass or wearable devices, leave the remaining options disabled before clicking *Next.*

On the final screen, enter *AndroidSampleActivity* for both the Activity Name and Title fields and name the layout *activity_android_sample*.

## 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 *Master/Detail Flow* option will be covered in a later chapter. For the purposes of this example, however, simply select the option to create a *Blank Activity* before clicking *Next.*

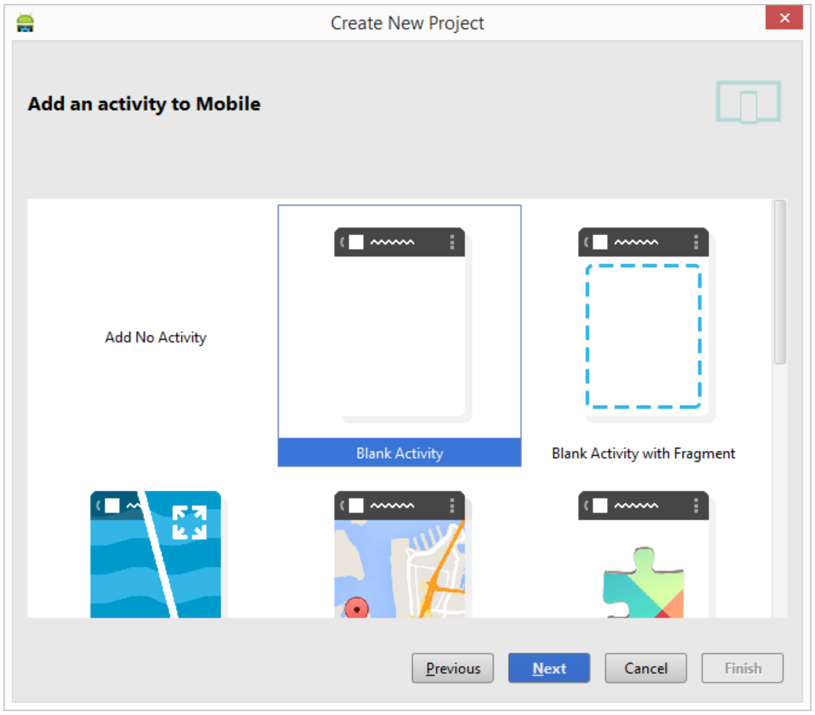
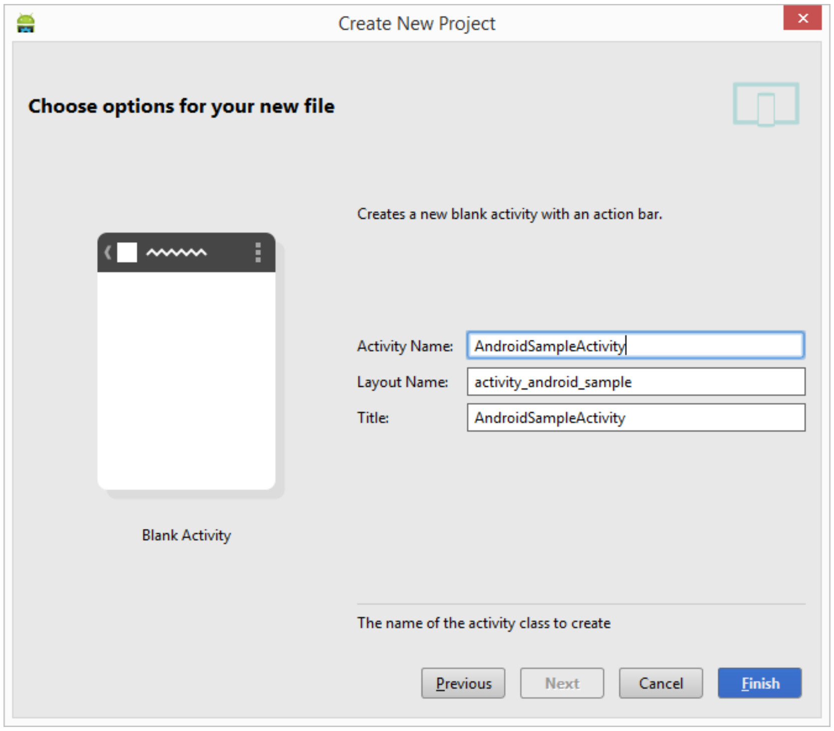


Figure 3-3

With the Blank Activity option selected, click *Next.* On the final screen (Figure 3-4) name the activity and title *AndroidSampleActivity*. The activity will consist of a single user interface screen layout which, for the purposes of this example, should be named *activity_android_sample* as shown in Figure 3-4:



Choose options for your new file

Creates a new blank activity with an action bar.

Activity Name: AndroidSampleActivity

Layout Name: activity_android_sample

Title: AndroidSampleActivity

The name of the activity class to create

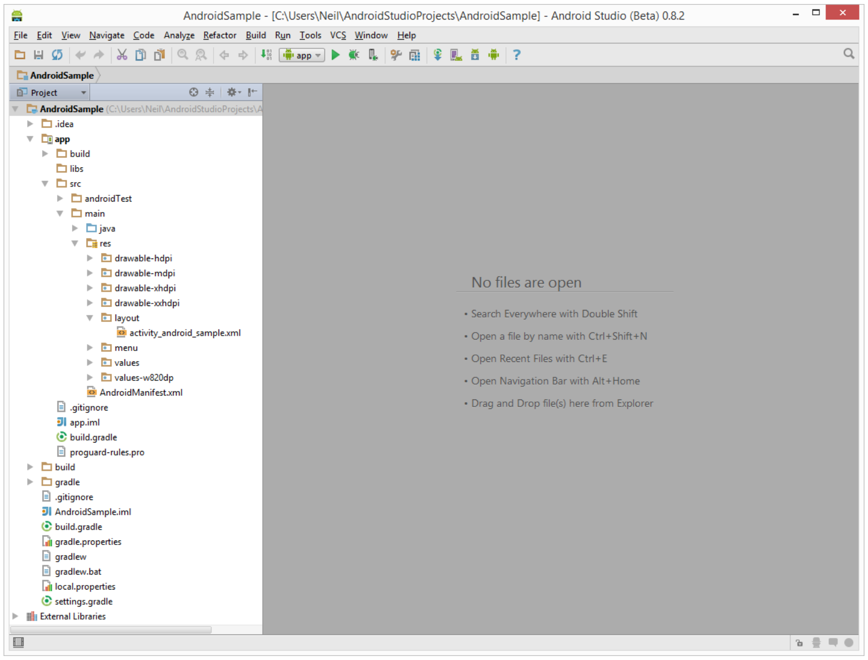
Figure 3-4

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

## 3.4 **Modifying the Example Application**

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

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. This essentially mirrors the directory hierarchy and files located in the project folder on the local file system of the development computer:



AndroidSample - [C:\Users\Neil\AndroidStudioProjects\AndroidSample] - Android Studio (Beta) 0.8.2

No files are open

- Search Everywhere with Double Shift
- Open a file by name with Ctrl+Shift+N
- Open Recent Files with Ctrl+E
- Open Navigation Bar with Alt+Home
- Drag and Drop file(s) here from Explorer

**Figure 3-5**

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 *AndroidSample -> app -> src -> res -> layout* in the project file hierarchy. Using the Project tool window, locate this file as illustrated in Figure 3-6:

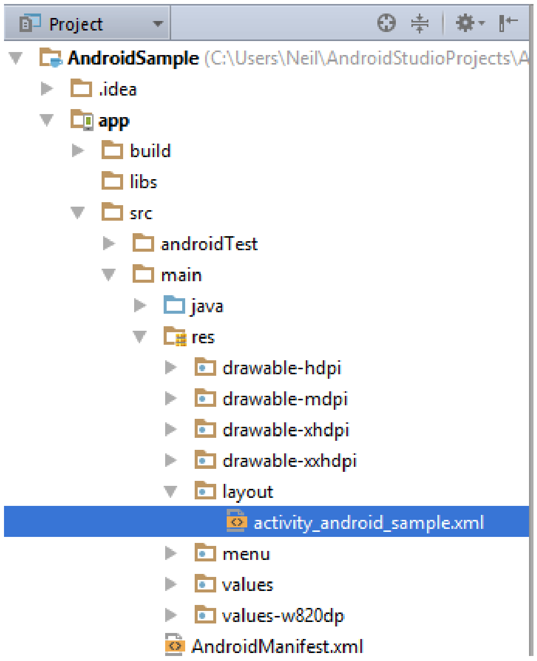
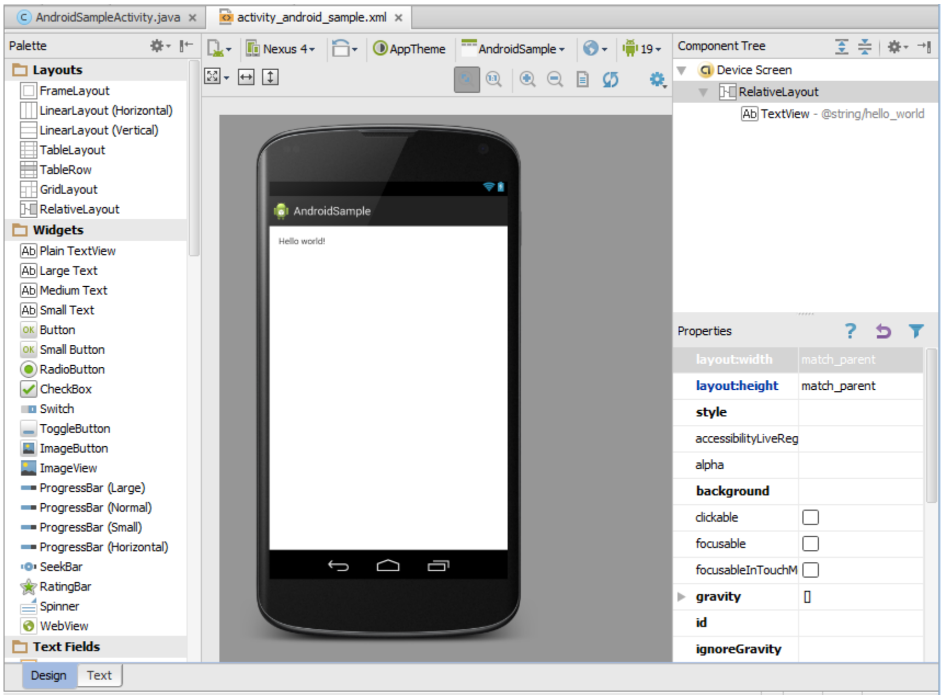


**Figure 3-6**

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



| | |
|---|---|
| layout:width | match_parent |
| layout:height | match_parent |
| style | |
| accessibilityLiveReg | |
| alpha | |
| background | |
| clickable | ☐ |
| focusable | ☐ |
| focusableInTouchM | ☐ |
| gravity | [] |
| id | |
| ignoreGravity | |

Figure 3-7

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

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

As can be seen in the device screen, the 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 different layouts that provide different levels of control over how visual user interface components are positioned and managed on the screen. Though it is difficult to tell from looking at the visual representation of the user interface, the current design has been created using a RelativeLayout. This can be confirmed by reviewing the information in the *Component Tree* panel which, by default, is located in the upper right hand corner of the Designer panel and is shown in Figure 3-8:

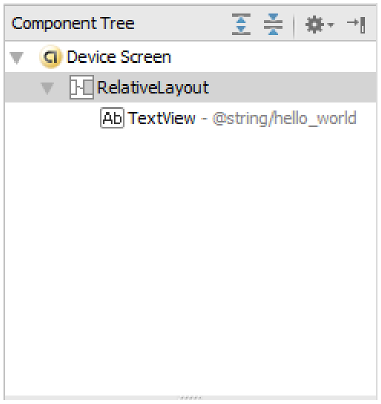


Figure 3-8

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

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

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

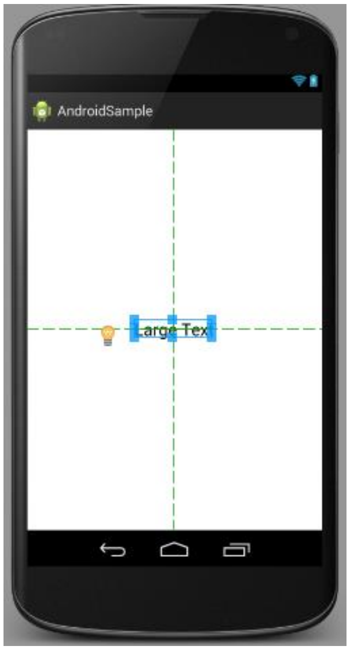


Figure 3-9

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

The next step is to change the text that is currently displayed by the TextView component. Double click on the object in the design layout to display the text and id editing panel as illustrated in Figure 3-10. Within the panel, change the text property from “Large Text” to “Welcome to Android Studio”.

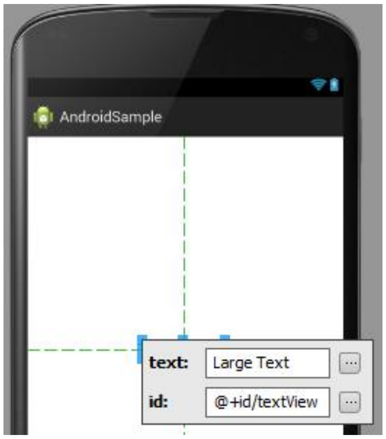


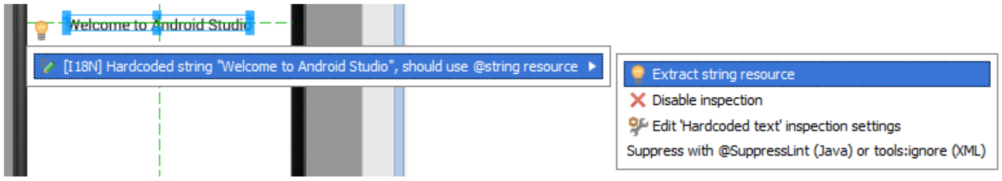
Figure 3-10

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

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

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

Click on the arrow to the right of the warning message to display the menu of possible solutions (Figure 3-11).


Welcome to Android Studio
Extract string resource
Disable inspection
Edit 'Hardcoded text' inspection settings
Suppress with @SuppressLint (Java) or tools:ignore (XML)

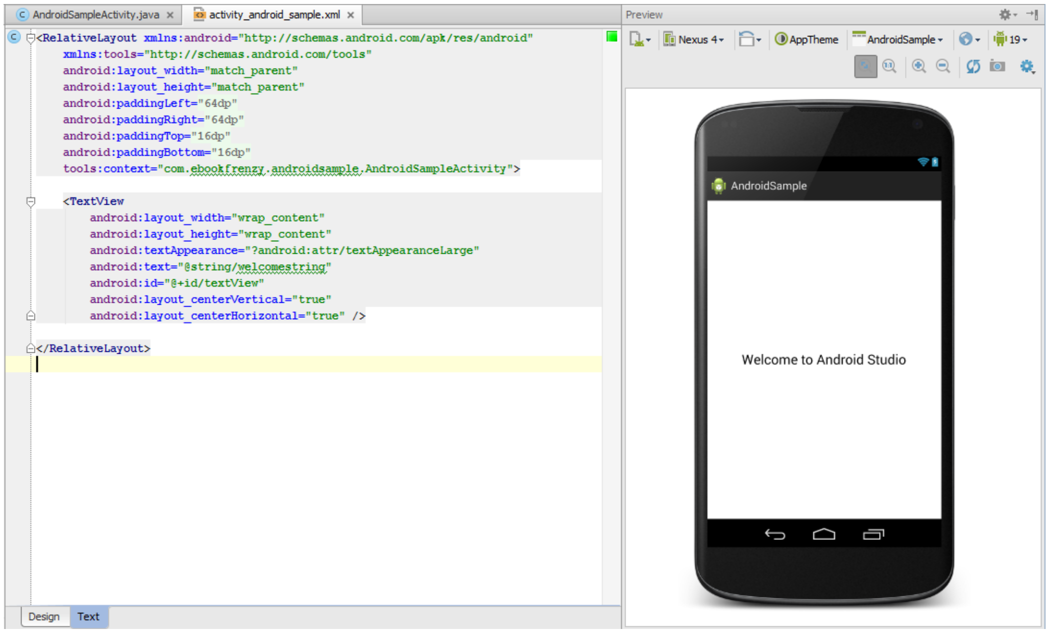
[I18N] Hardcoded string "Welcome to Android Studio", should use @string resource

Figure 3-11

From the menu, select the *Extract string resource* option to display the *Extract Resource* dialog. In this dialog, enter *welcomestring* into the *Resource name:* field before clicking on *OK.* The string is now stored as a resource in the *src -> res -> values -> strings.xml* file.

## 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 *activity_android_sample.xml* file using the UI Designer tool. In fact, all that the Designer was doing was providing a user-friendly way to edit the underlying XML content of the file. In practice, there is no reason why you cannot modify the XML directly in order to make user interface changes and, in some instances, this may actually be quicker than using the Designer tool. At the bottom of the Designer panel are two tabs labeled *Design* and *Text* respectively. To switch to the XML view simply select the *Text* tab as shown in Figure 3-12:


AndroidSampleActivity.java
activity_android_sample.xml
Preview
Nexus 4
AppTheme
AndroidSample
19
AndroidSample
Welcome to Android Studio
Design
Text

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context="com.ebookfrenzy.androidsample.AndroidSampleActivity">

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

</RelativeLayout>
```

Figure 3-12

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

One of the more powerful features of Android Studio can be found to the right hand side of the XML editing panel. This is the Preview panel and shows the current visual state of the layout. As changes are made to the XML layout, these will be reflected in the preview panel. To see this in action, modify the XML layout to change the background color of the RelativeLayout to a shade of red as follows:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".AndroidSampleActivity"
    android:background="#ff2438">

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

</RelativeLayout>
```

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

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

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

```
<resources>

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

</resources>
```

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

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

## 3.6 **Previewing the Layout**

So far in this chapter, the layout has only been previewed on a representation of the Nexus 4 device. As previously discussed, the layout can be tested for other devices by making selections from the device menu in the toolbar across the top edge of the Designer panel. Another useful option provided by this menu is *Preview All Screen Sizes* which, when selected, shows the layout in all currently configured device configurations as demonstrated in Figure 3-13:

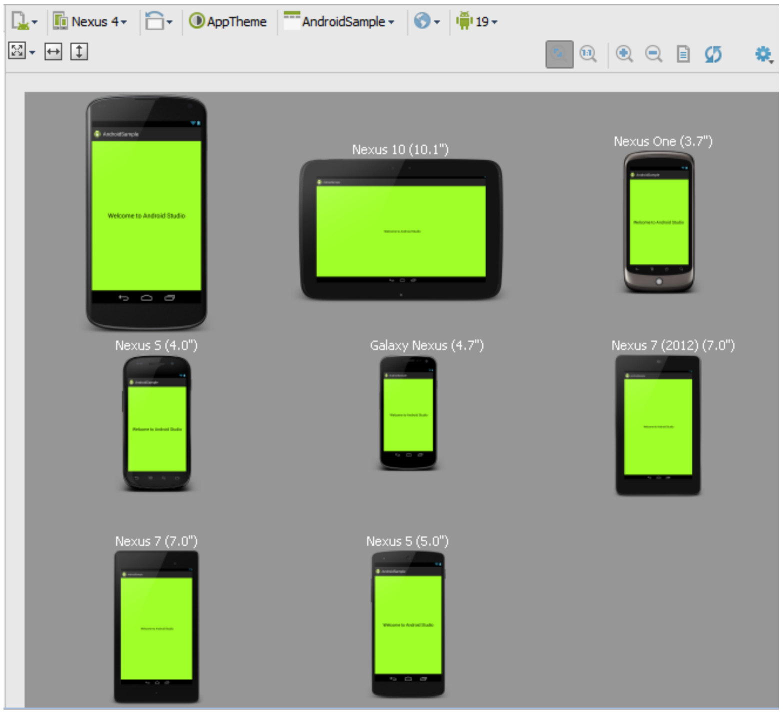


Figure 3-13

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

## 3.7 **Summary**

Whilst 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 UI Designer 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.

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

Chapter 4

# 4. A Tour of the Android Studio User Interface

Whilst 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.

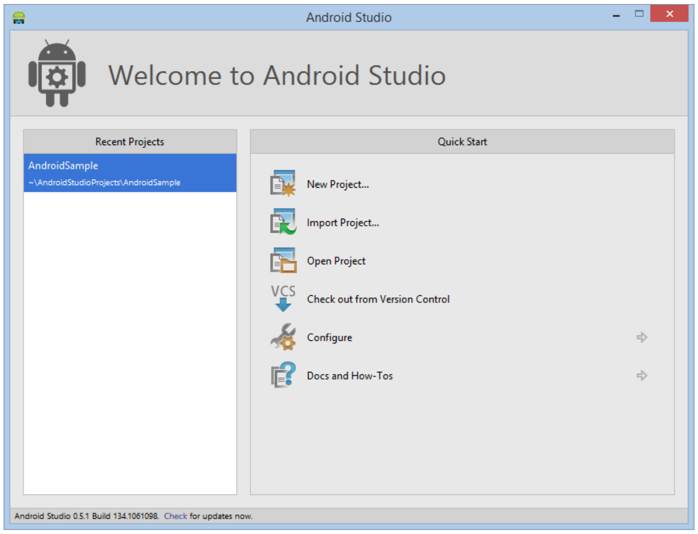

Android Studio
Welcome to Android Studio
Recent Projects
AndroidSample
~\AndroidStudioProjects\AndroidSample
Quick Start
New Project...
Import Project...
Open Project
Check out from Version Control
Configure
Docs and How-Tos
Android Studio 0.5.1 Build 134.1061098. Check for updates now.

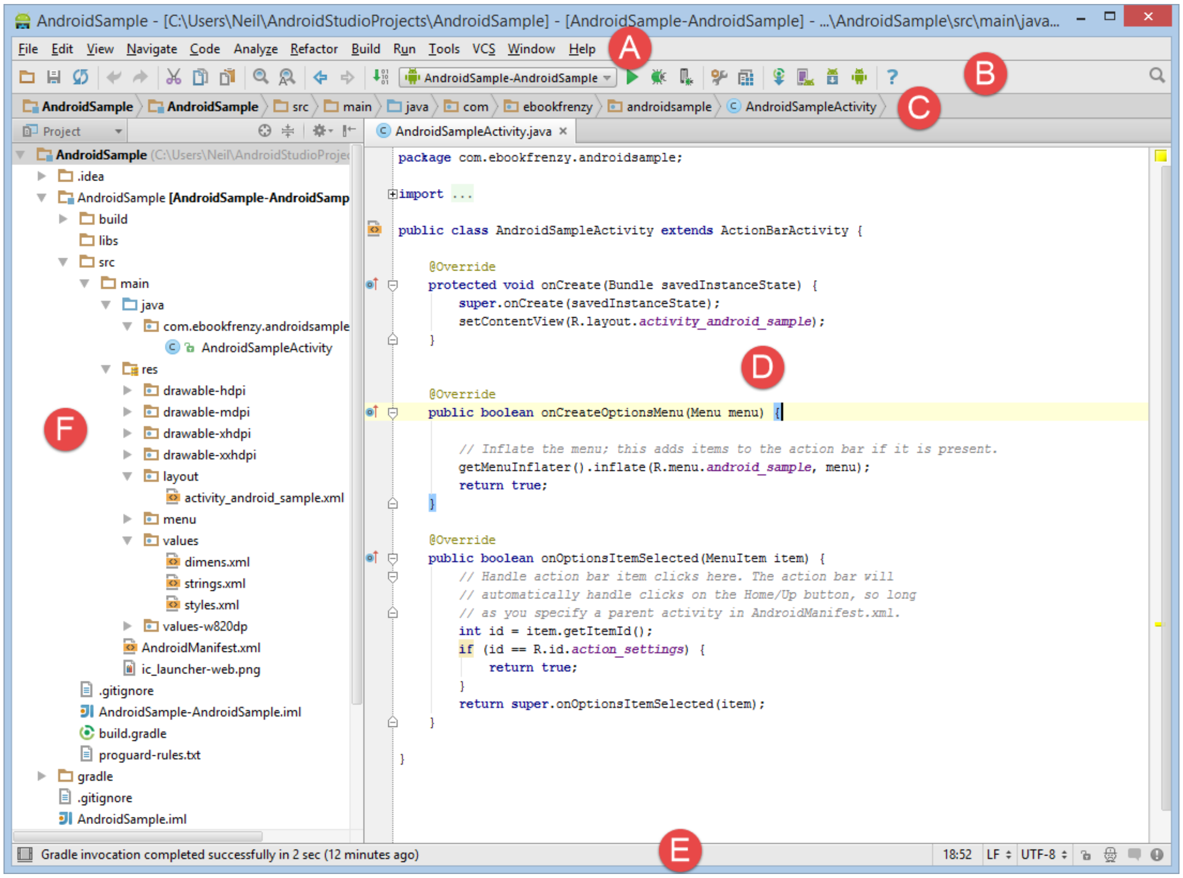
Figure 4-1

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* option 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.

Finally, the status bar along the bottom edge of the window provides information about the version of Android Studio currently running, along with a link to check if updates 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.


AndroidSample - [C:\Users\Neil\AndroidStudioProjects\AndroidSample] - [AndroidSample-AndroidSample] - ...\AndroidSample\src\main\java...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
A
AndroidSample-AndroidSample
B
AndroidSample AndroidSample src main java com ebookfrenzy androidsample AndroidSampleActivity
C
Project
AndroidSampleActivity.java
AndroidSample (C:\Users\Neil\AndroidStudioProje
.idea
AndroidSample [AndroidSample-AndroidSamp
build
libs
src
main
java
com.ebookfrenzy.androidsample
AndroidSampleActivity
res
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
activity_android_sample.xml
menu
values
dimens.xml
strings.xml
styles.xml
values-w820dp
AndroidManifest.xml
ic_launcher-web.png
.gitignore
AndroidSample-AndroidSample.iml
build.gradle
proguard-rules.txt
gradle
.gitignore
AndroidSample.iml
D
E
F
18:52 LF UTF-8

```
package com.ebookfrenzy.androidsample;

import ...

public class AndroidSampleActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_android_sample);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.android_sample, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

Gradle invocation completed successfully in 2 sec (12 minutes ago)

Figure 4-2

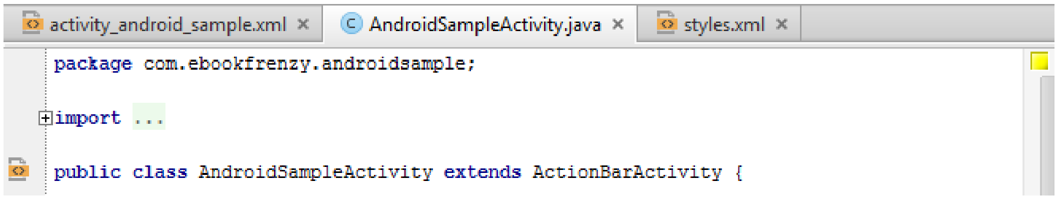
The various elements of the main window can be summarized as follows:

**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 Designer 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.



```
package com.ebookfrenzy.androidsample;

import ...

public class AndroidSampleActivity extends ActionBarActivity {
```

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 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.

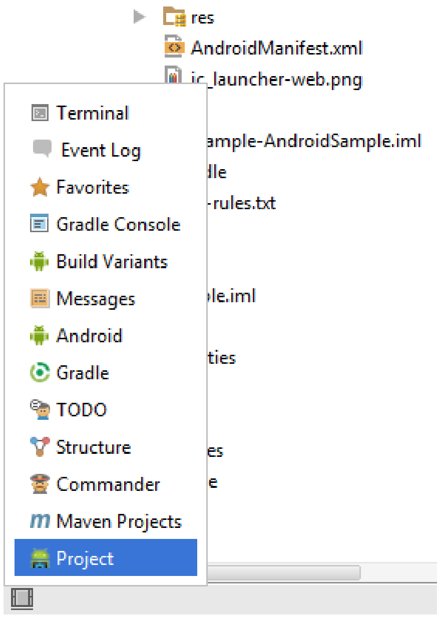


Figure 4-4

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.

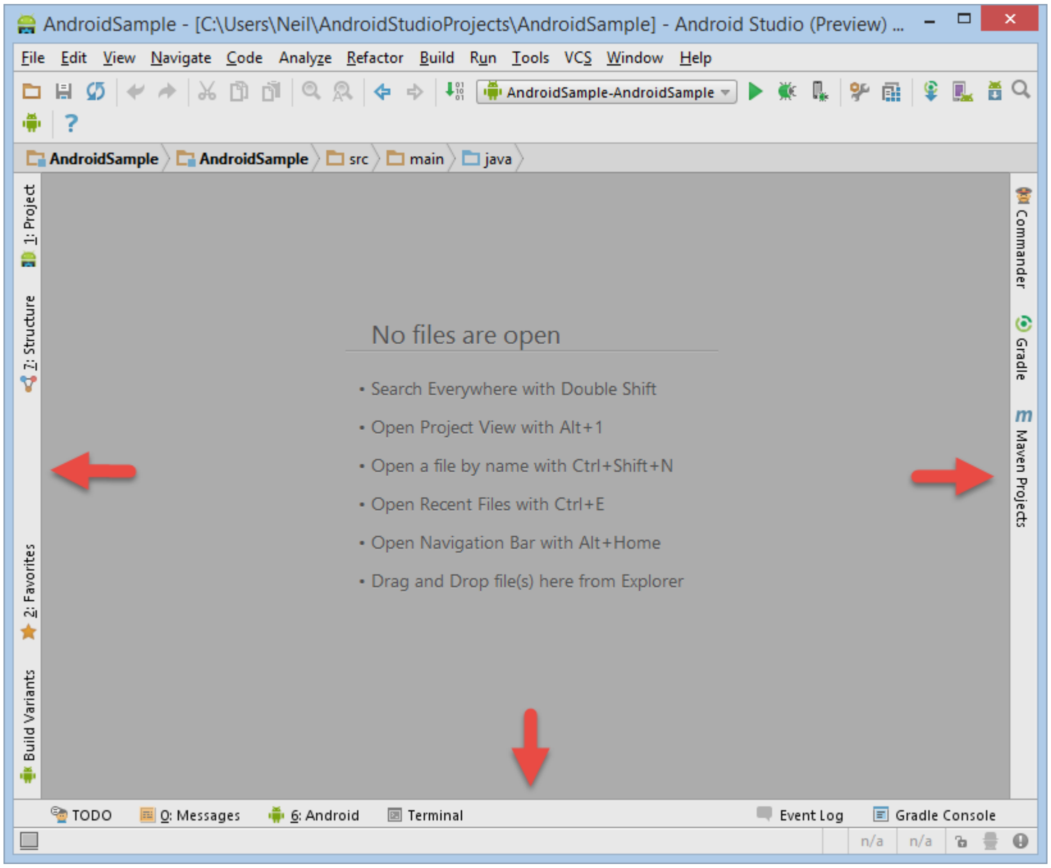

AndroidSample - [C:\Users\Neil\AndroidStudioProjects\AndroidSample] - Android Studio (Preview) ...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AndroidSample-AndroidSample
AndroidSample AndroidSample src main java
1: Project
7: Structure
2: Favorites
Build Variants
Commander
Gradle
Maven Projects
No files are open
• Search Everywhere with Double Shift
• Open Project View with Alt+1
• Open a file by name with Ctrl+Shift+N
• Open Recent Files with Ctrl+E
• Open Navigation Bar with Alt+Home
• Drag and Drop file(s) here from Explorer
TODO
0: Messages
6: Android
Terminal
Event Log
Gradle Console
n/a
n/a

Figure 4-5

Clicking on a button will display the corresponding tool window whilst 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 Mac OS X) 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.

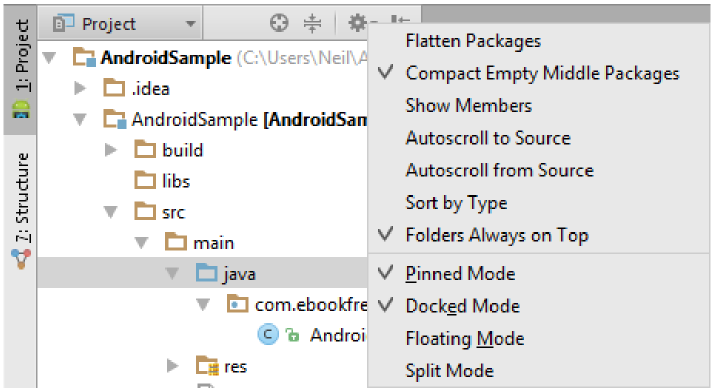


Figure 4-6

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 window 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.

**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 and navigating to the *TODO* page listed under *IDE Settings*.

**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 a runnable applications) and can be useful for identifying the causes of build problems when compiling application projects.

**Android** – The Android tool window provides access to the Android debugging system. Within this window tasks such as monitoring log output from a running application, taking screenshots and videos of the application, stopping a process and performing basic debugging tasks can be performed.

**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, whilst on Linux and Mac OS X 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 window tool.

**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.

**Maven Projects** – Maven is a project management and build system designed to ease the development of complex Java based projects and overlaps in many areas with the functionality provided by Gradle. Google has chosen Gradle as the underlying build system for Android development, so unless you are already familiar with Maven or have existing Maven projects to import, your time will be better spent learning and adopting Gradle for your projects. The Maven projects tool window can be used to add, manage and import Maven based projects within Android Studio.

**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.

**Commander** – The Commander window tool can best be described as a combination of the Project and Structure tool windows, allowing the file hierarchy of the project to be traversed and for the various elements that make up classes to be inspected and loaded into the editor or designer windows.

**Designer** – Available when the UI Designer is active, this tool window provides access to the designer's Component Tree and Properties panels.

## 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 -> Default 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).

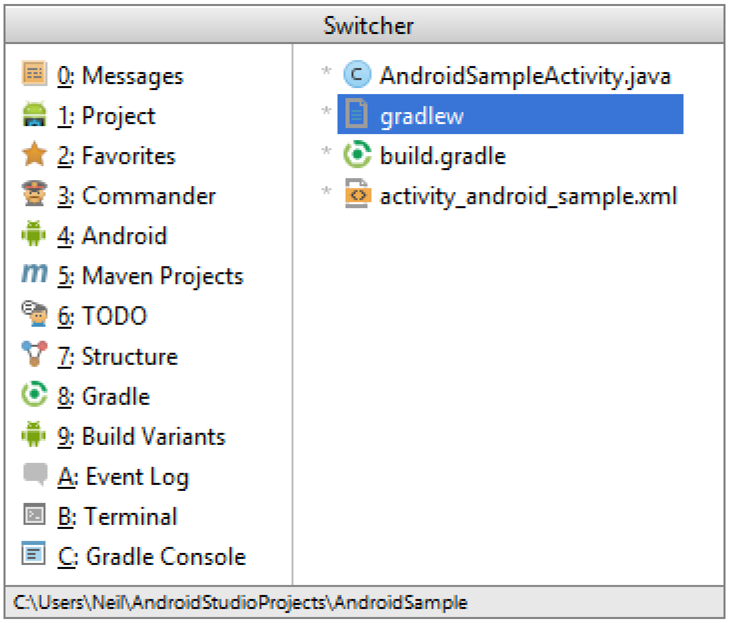


Figure 4-7

Once displayed, the switcher will remain visible for as long the Ctrl key remains depressed. Repeatedly tapping the Tab key whilst holding down the Ctrl key will cycle through the various selection options, whilst 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 Mac OS X). Once displayed, either the mouse pointer can be used to select an option or, alternatively, the keyboard arrow keys can be used to scroll through the file name and tool window options. Pressing the Enter key will select the currently highlighted item.

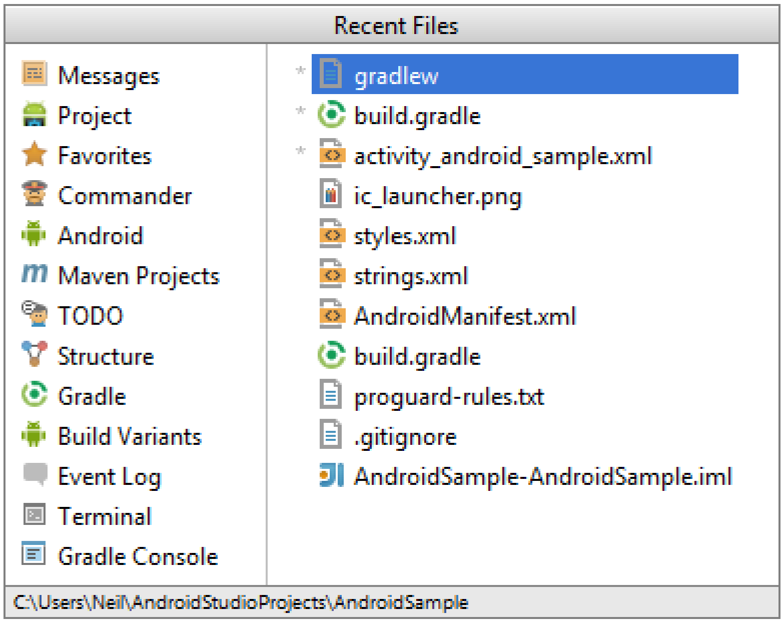
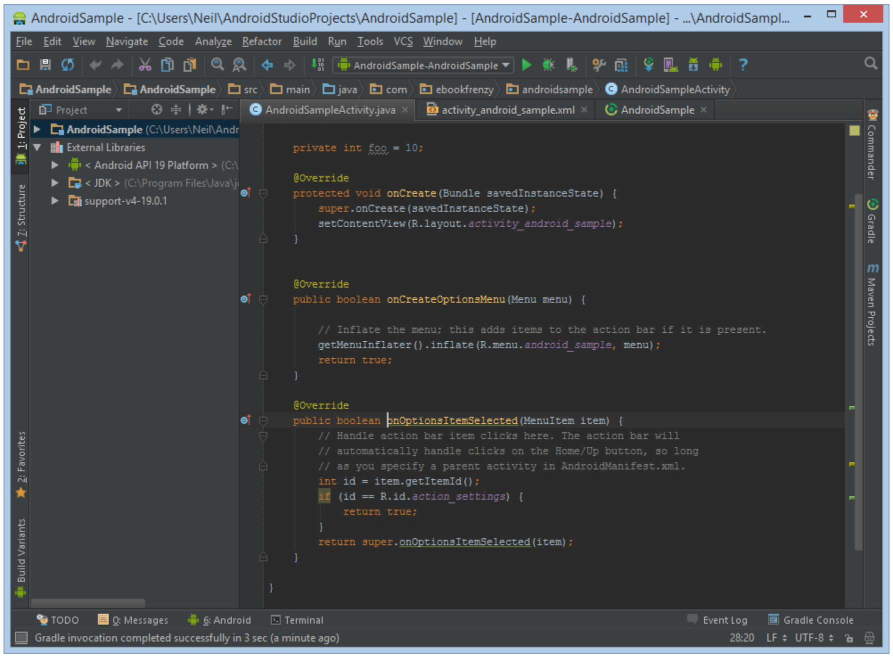


Figure 4-8

## 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 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 currently available consist of IntelliJ, Windows and Darcula. Figure 4-9 shows an example of the main window with the Darcula theme selected:


AndroidSample - [C:\Users\Neil\AndroidStudioProjects\AndroidSample] - [AndroidSample-AndroidSample] - ...\AndroidSampl...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AndroidSample-AndroidSample
AndroidSample AndroidSample src main java com ebookfrenzy androidsample AndroidSampleActivity
Project
AndroidSampleActivity.java
activity_android_sample.xml
AndroidSample
AndroidSample (C:\Users\Neil\Andr
External Libraries
< Android API 19 Platform > (C:\
< JDK > (C:\Program Files\Java\j
support-v4-19.0.1
1: Project
7: Structure
2: Favorites
Build Variants
Commander
Gradle
Maven Projects
TODO
0: Messages
6: Android
Terminal
Event Log
Gradle Console
28:20 LF UTF-8

```
    private int foo = 10;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_android_sample);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.android_sample, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

```
Gradle invocation completed successfully in 3 sec (a minute ago)
```

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.

Chapter 5

# 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 specification 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 7 tablet 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 7 device.

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.

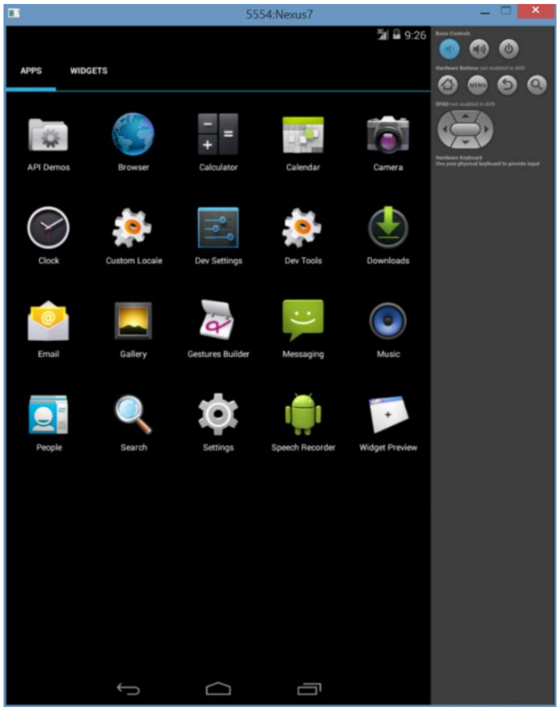


Figure 5-1

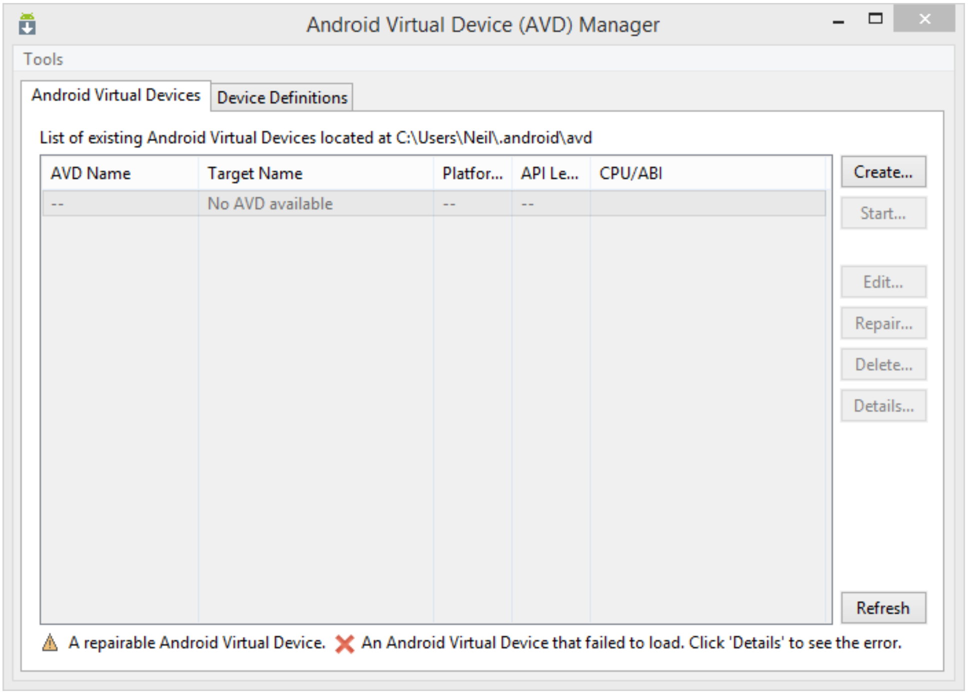
## 5.2 **Creating a New AVD**

In order to test the behavior of an application, 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. Alternatively, the tool may be launched from a terminal or command-line prompt using the following command:

```
android avd
```

Once launched, the tool will appear as outlined in Figure 5-2. Assuming a new Android SDK installation, no AVDs will currently be listed:



List of existing Android Virtual Devices located at C:\Users\Neil\.android\avd

| AVD Name | Target Name | Platfor... | API Le... | CPU/ABI |
|---|---|---|---|---|
| -- | No AVD available | -- | -- | |

⚠ A repairable Android Virtual Device. ✖ An Android Virtual Device that failed to load. Click 'Details' to see the error.

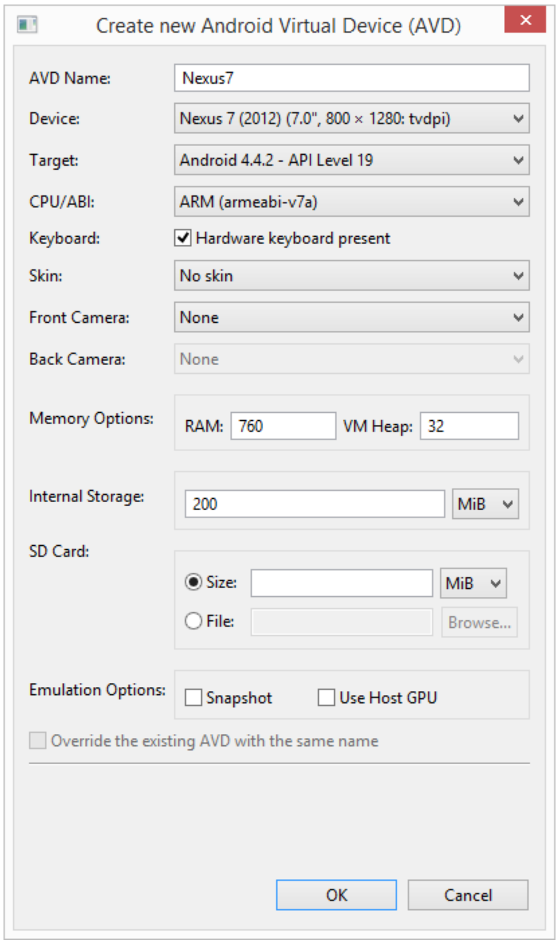
Figure 5-2

Begin the AVD creation process by clicking on the *New…* button in order to invoke the *Create a New Android Virtual Device (AVD)* dialog. Within the dialog, perform the following steps to create a first generation Nexus 7 compatible emulator:

1. Enter a descriptive name (for example *Nexus7*) into the name field. Note that spaces and other special characters are not permitted in the name.
2. Set the *Device* menu to *Nexus 7 (2012) (7.0", 800 x 1280: tvhdpi).*
3. Set the *Target* menu to *Android 4.4 – API Level 19*.
4. Set the *CPU/ABI* menu to *ARM (armeabi-v7a)*.
5. Set the Skin menu to "no skin" so that the device is displayed without simulated hardware controls.
6. Leave the default *RAM* value in *Memory Options* and the *Internal Storage* value unchanged from the default settings. Keep in mind however, that it may be necessary to reduce the RAM value below 768M on Windows systems with less available memory.
7. If the host computer contains a web cam the *Front Camera:* emulation may be configured to use this camera. Alternatively, an emulated camera may be selected. If camera functionality is not required by the application, simply leave this set to *None*.

Whether or not you enable the *Hardware keyboard present* option is a matter of personal preference. When the hardware keyboard option is selected, it will be possible to use the physical keyboard on the system on which the emulator is running. As such, the Android software keyboard will not appear within the emulator when text input is required by a running application.

Note that it may also be possible to speed the performance of the emulator by enabling the *Use Host GPU* option. In the event that the emulator crashes during startup when this option is selected, edit the virtual device properties and disable this option. Figure 5-3 illustrates the dialog with the appropriate settings implemented for a Nexus 7 emulator. Once the configuration settings are complete, click on the *OK* button.



AVD Name: Nexus7
Device: Nexus 7 (2012) (7.0", 800 × 1280: tvdpi)
Target: Android 4.4.2 - API Level 19
CPU/ABI: ARM (armeabi-v7a)
Keyboard: [✓ Hardware keyboard present]
Skin: No skin
Front Camera: None
Back Camera: None
Memory Options: RAM: 760 VM Heap: 32
Internal Storage: 200 MiB
SD Card: [● Size:] MiB ○ File: Browse...
Emulation Options: ☐ Snapshot ☐ Use Host GPU
☐ Override the existing AVD with the same name

Figure 5-3

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 *Edit…* button.

## 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 *Start…* button followed by *Launch* in the resulting *Launch Options* dialog. The emulator will appear in a new window and, after a short period of time, the “android” logo will appear in the center of the screen. The first time the emulator is run, it can take up to 10 minutes for the emulator to fully load and start. On subsequent invocations, this will typically reduce to a few minutes. 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 is already running and attach to it when applications are launched, thereby saving considerable amounts of startup time.

Another option when using the emulator is to enable the *Snapshot* option in the AVD settings screen. This option, which can only be used when the *Use Host GPU* option is disabled, enables the state of an AVD instance to be saved and reloaded next time it is launched. This can result in an emulator startup time of just a few seconds.

When using snapshots, make sure that the *Launch from snapshot* and *Save to snapshot* options are both enabled in the Launch Options dialog. The first time an emulator instance is launched with snapshots enabled, the AVD will start in the normal manner. On closing the emulator window, a slight delay will occur while the state of the AVD is saved. Next time the emulator is a launched it will load more quickly and revert to the previously saved state.

To revert the AVD to its original, pre-snapshot state, simply a launch it with the *Launch from snapshot* option disabled.

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…* menu option or use the Shift+F10 keyboard shortcut:

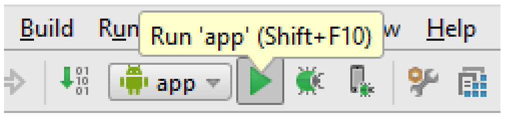


Figure 5-4

By default, Android Studio will respond to the run request by displaying the *Choose Device* 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 Nexus7 AVD as