Android 4.2 App Development

Essentials

Android 4.2 App Development Essentials

Android 4.2 App Development Essentials – First Edition

ISBN-13: 978-0-9860273-4-5

© 2013 Neil Smyth. All Rights Reserved.

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

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

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

Rev 1.0

Table of Contents

Chapter 1

1. Introduction

The goal of this book is to teach the skills necessary to develop Android based applications using the Eclipse Integrated Development Environment (IDE) and the Android 4.2 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 introduction to the architecture of Android is followed by an in-depth look at the design of Android applications and user interfaces. 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.

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

Assuming you already have some Java programming experience, are ready to download Eclipse 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 Eclipse project files for the examples contained in this book are available for download at:

http://www.ebookfrenzy.com/direct/android42/index.php

Once the file has been downloaded and unzipped, the samples may be imported into an existing Eclipse workspace by selecting the Eclipse *File -> Import…* menu option and choosing the *Android -> Existing Android Code Into Workspace* category. When prompted, select the folder containing the sample project folders as the *Root Directory* before choosing the sample projects to be imported from the resulting list.

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/android42.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 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), the Eclipse Integrated Development Environment (IDE) and the appropriate Android Software Development Kit (SDK). In addition to these steps, it will also be necessary to install the Eclipse Android Development Tool (ADT) Plug-in.

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
- 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)

Both the Eclipse IDE and Android SDK were 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_37"
Java(TM) SE Runtime Environment (build 1.6.0_37-b06-434-11M3909)
Java HotSpot(TM) 64-Bit Server VM (build 20.12-b01-434, mixed mode)
```

In the event that Java is not installed, issuing the “java” command in the terminal window should initiate the JDK installation process.

2.2.3 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-7u10-linux-x64.rpm*, the commands to perform the installation would read as follows:

```
su
rpm -ihv jdk-7u10-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_10* the following would need to be added to your $PATH environment variable:

```
/home/demo/java/jdk1.7.0_10/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_10/bin:$PATH
```

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

2.3 Downloading the Android Developer Tools (ADT) Bundle

Most of the work involved in developing applications for Android will be performed using the Eclipse Integrated Development Environment (IDE). If you are already using Eclipse to develop for other platforms, then the Android Developer Tools (ADT) plug-in can be integrated into your existing Eclipse installation (a topic covered later in this chapter). If, on the other hand, you are entirely new to Eclipse based development, the most convenient path to take is to install a package known as the *ADT Bundle*. This bundle includes many of the tools necessary to begin developing Android applications in a single download.

The ADT Bundle may be downloaded from the following web page:

https://developer.android.com/sdk/index.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 the SDK ADT Bundle 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, the target architecture of your computer system (32-bit or 64-bit) and click on the download button. Note that your choice of 32-bit or 64-bit should match the architecture chosen for the JDK installation. Attempting to run a 64-bit ADT bundle using a 32-bit JDK, for example, will result in errors when attempting to launch Eclipse.

2.4 Installing the ADT Bundle

The ADT Bundle is downloaded as a compressed ZIP archive file which must be unpacked to complete the installation process. The exact steps to achieve this differ depending on the operating system.

2.4.1 Installation on Windows

Locate the downloaded ADT Bundle zip file in a Windows Explorer window, right-click on it and select the *Extract All…* menu option. In the resulting dialog, choose a suitable location into which to unzip the file

before clicking on the *Extract* button. When choosing a suitable location, keep in mind that the extraction will create a sub-folder in the chosen location named either *adt-bundle-windows-x86* or *adt-bundle-windows-x86_64* containing the bundle packages.

Once the extraction is complete, navigate in Windows Explorer to the directory containing the ADT bundle, move into the *eclipse* sub-folder and double click on the *eclipse* executable to start the Eclipse IDE environment. For easier future access, right click on the eclipse executable and select *Pin to Taskbar* from the resulting menu.

It is possible that Windows will display a Security Warning dialog before Eclipse will launch stating that the publisher could not be verified. In the event that this warning appears, uncheck the "Always ask before opening this file" option before clicking the *Run* button. Once invoked, Eclipse will prompt for the location of the workspace. All projects will be stored by default into this folder. Browse for a suitable location, or choose the default offered by Eclipse and click on *OK*.

2.4.2 Installation on Mac OS X

On Mac OS X systems, open a terminal window, change directory to the location where Eclipse is to be installed and execute the following command:

```
unzip /<path to package>/<package name>.zip
```

For example, assuming a package file named *adt-bundle-mac-x86_64.zip* has been downloaded to */home/demo/Downloads*, the following command would be needed to install Eclipse:

```
unzip /home/demo/Downloads/adt-bundle-mac-x86_64.zip
```

Note that the bundle will be installed into a sub-directory named *adt-bundle-mac-x86_64.* Assuming, therefore, that the above command was executed in */Users/demo*, the software packages will be unpacked into */Users/demo/adt-bundle-mac-x86_64.* Within this directory, the files comprising the Eclipse IDE are installed in a sub-directory named *eclipse.*

Using the Finder tool, navigate to the *eclipse* sub-directory of the ADT bundle installation directory and double click on the *eclipse* executable to launch the application. For future easier access to the tool, simply drag the eclipse icon from the Finder window and drop it onto the dock.

2.4.3 Installation on Linux

On Linux systems, open a terminal window, change directory to the location where Eclipse is to be installed and execute the following command:

```
unzip /<path to package>/<package name>.zip
```

For example, assuming a package file named *adt-bundle-linux-x86.zip* has been downloaded to */home/demo/Downloads*, the following command would be needed to install Eclipse:

```
unzip /home/demo/Downloads/adt-bundle-linux-x86.zip
```

Note that the bundle will be installed into a sub-directory named either *adt-bundle-linux-x86* or adt-*bundle-linux-x86_64* depending on whether the 32-bit or 64-bit edition was downloaded. Assuming, therefore, that the above command was executed in */home/demo*, the software packages will be unpacked into */home/demo/adt-bundle-linux-x86.* Within this directory, the files comprising the Eclipse IDE are installed in a sub-directory named *eclipse.*

To launch Eclipse, open a terminal window, change directory to the *eclipse* sub-directory of the ADT bundle installation directory and execute the following command:

```
./eclipse
```

Once invoked, Eclipse will prompt for the location of the workspace. All projects will be stored by default into this folder. Browse for a suitable location, or choose the default offered by Eclipse and click on *OK*.

Having verified that the Eclipse IDE is installed correctly, keep Eclipse running so that it can be used to install additional Android SDK packages.

2.5 Installing the Latest Android SDK Packages

The steps performed so far have installed Java, the Eclipse 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 Eclipse tool by selecting the *Window -> Android SDK Manager* menu option. Once invoked, the SDK Manager tool will appear as illustrated in Figure 2-1:

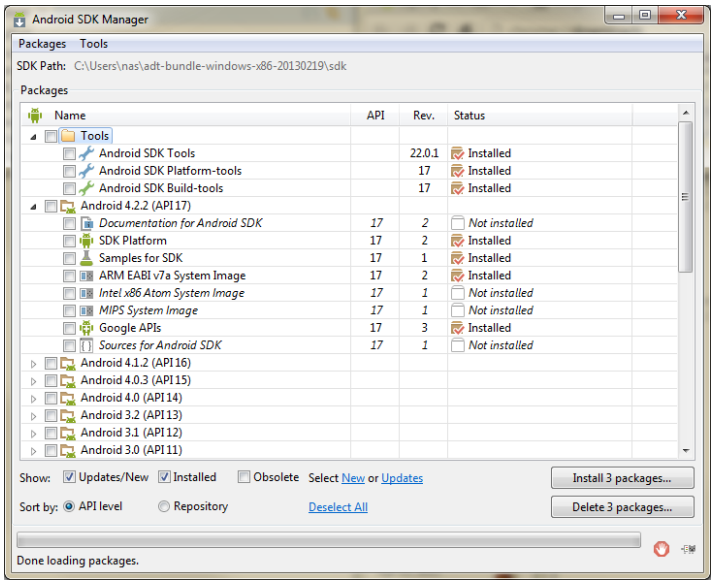

Figure 2-1

Once the SDK Manager is running, return to the main Eclipse window and select the *File -> Exit* menu option to exit from the Eclipse environment. This will leave the Android SDK Manager running whilst ensuring that the Eclipse session does not conflict with the installation process.

Begin by checking that the *SDK Path:* setting at the top of the SDK Manager window matches the location into which the ADT Bundle package was unzipped. If it does not, relaunch Eclipse and select the *Window -> Preferences* option. In the *Preferences* dialog, select the *Android* option from the left hand panel and change the *SDK Location* setting so that it references the *sdk* sub-folder of the directory into which the ADT Bundle was unzipped before clicking on *Apply* followed by *OK*.

Within the Android SDK Manager, make sure that the check boxes next to the following packages are selected:

- Tools > Android SDK Tools
- Tools > Android SDK Platform-tools
- SDK Platform Android 4.2.2 API 17 > SDK Platform
- SDK Platform Android 4.2.2 API 17 > ARM EABI v7a System Image
- Extras > Android Support Library

With the appropriate package selections made (and assuming these packages were not already installed), 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.

2.6 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 Eclipse 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_adt_installation>* represents the file system location into which the ADT bundle was installed):

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

The steps to achieve this 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 ADT bundle was installed into */Users/demo/adt-bundle-windows-x86_64*, the following would be appended to the end of the current Path value:

```
;C:\Users\demo\adt-bundle-windows-x86_64\sdk\platform-
tools;C:\Users\demo\adt-bundle-windows-x86_64\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 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. 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.

2.6.3 Linux

On Linux this will involve once again editing the *.bashrc* file. Assuming that the bundle package was installed into */home/demo/adt-bundle-linux-x86*, the export line in the *.bashrc* file would now read as follows:

```
export PATH=/home/demo/java/jdk1.7.0_10/bin:/home/demo/adt-bundle-linux-
x86/sdk/platform-tools:/home/demo/adt-bundle-linux-x86/sdk/tools:$PATH
```

2.6.4 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 */Users/demo/adt-bundle-mac-x86_64*, 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/adt-bundle-mac-x86_64/sdk/tools
/Users/demo/adt-bundle-mac-x86_64/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 the ADT

From time to time new versions of the Android ADT and SDK are released. New versions of the SDK are installed using the Android SDK Manager. When new versions of the SDK have been installed on your system the ADT will also often need to be updated to a matching version. The latest version of the ADT can be installed by selecting the Eclipse *Help -> Install New Software* menu option. When prompted, enter the following URL and a suitable name for the update (the choice of name is not important):

https://dl-ssl.google.com/android/eclipse/

Having entered the required information Eclipse will list any available updates. If updates are listed, simply proceed with the installation process. Once complete, restart Eclipse to use the latest version of the ADT.

2.8 Adding the ADT Plugin to an Existing Eclipse Integration

The steps outlined so far in this chapter have assumed that the Eclipse IDE is not already installed on your system. In the event that you are already using Eclipse for Java based development, the appropriate Android development tools and SDKs can be added to this existing Eclipse installation. Eclipse editions with which the ADT Plugin is compatible are as follows:

- Eclipse IDE for Java Developers
- Eclipse Classic (versions 3.5.1 and higher)
- Eclipse IDE for Java EE Developers
- Eclipse for Mobile Developers

The ADT Plugin for Eclipse adds a range of Android specific features to what is otherwise a general-purpose Java edition of the Eclipse environment. To install this plugin, launch Eclipse and select the *Help –> Install New Software…* menu option. In the resulting window, click on the *Add…* button to display the *Add Repository* dialog. Enter "ADT Plugin" into the *Name* field and the following URL into the *Location* field:

```
https://dl-ssl.google.com/android/eclipse/
```

Click on the *OK* button and wait while Eclipse connects to the Android repository. Once the information has been downloaded, new items will be listed entitled *Developer Tools* and *NDK Plugins* as illustrated in Figure 2-2:

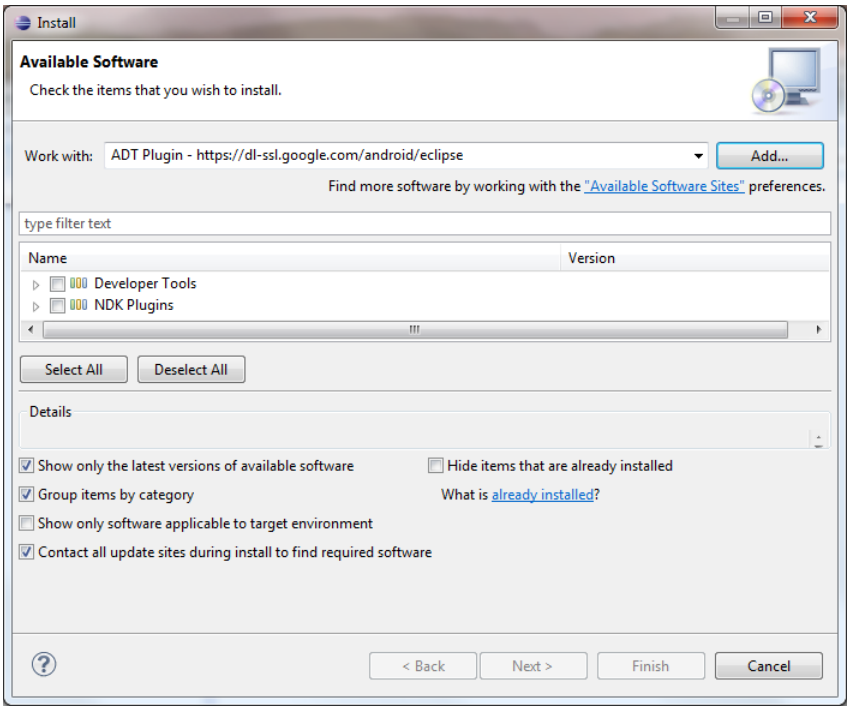

Figure 2-2

Select the checkbox next to the *Developer Tools* entry and click on the *Next >* button. After requirements and dependencies have been calculated by the installer, a more detailed list of the packages to be installed will appear. Once again click on the *Next >* button to proceed. On the subsequent licensing page, select the option to accept the terms of the agreements (assuming that you do, indeed, agree) and click on *Finish* to complete the installation. During the download and installation process, you may be prompted to confirm that you wish to install unsigned content. In the event that this happens, simply click on the option to proceed with the installation.

When the ADT Plugin installation is complete, a dialog will appear providing the option to restart Eclipse in order to complete the installation. Click on *Yes* and wait for the tool to exit and re-launch.

Upon restarting, the *Welcome to Android Development* dialog will appear as illustrated in Figure 2-3:

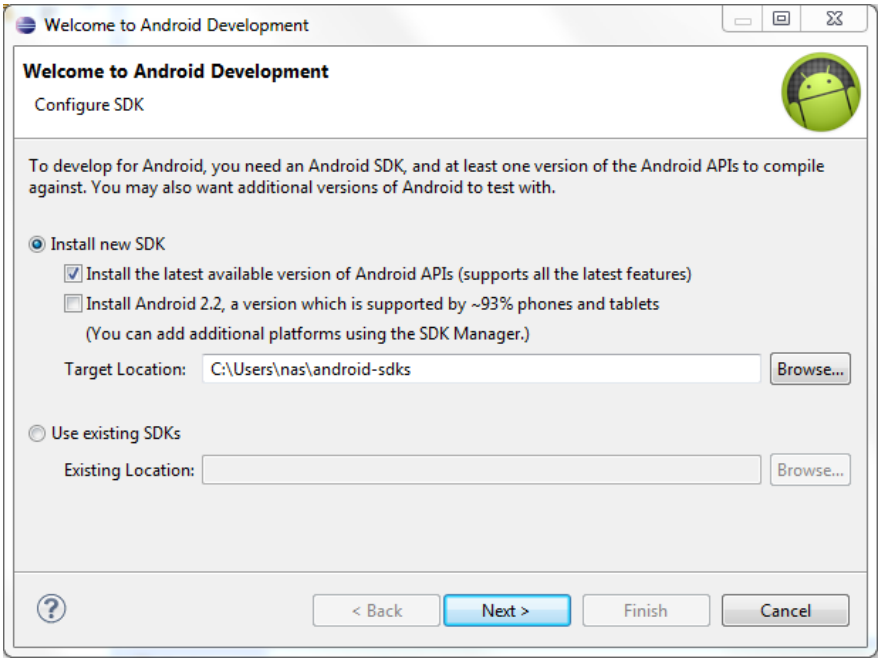

Figure 2-3

At this stage there is no existing SDK installed so the *Use Existing SDKs* choice is not a viable option. Unfortunately, the ADT Plugin does not provide the option at this point to install the SDKs of our choice so we will need to install the latest available SDK version. With this in mind, select the option to install the latest available version of the Android APIs. Make a note of the *Target Location* path and change it if you prefer the SDKs to be installed in a different location, then click *Next.* Choose whether to send usage information to Google, accept all the licensing terms and click on *Install.* The Android SDK Manager will now download and install the latest Android SDKs.

At this point, the Eclipse environment is ready to begin the development of Android applications.

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, Eclipse IDE and the Android ADT Plugin for Eclipse. 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 Android Virtual Device (AVD)

In the course of developing Android apps 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.

3.1 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 installation process outlined in the previous chapter, a number of emulator template definitions were 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 process type, memory capacity, screen size and density. Check the online developer documentation for your device to find out if emulator definitions are available for download and installation into the ADT environment.

When launched, an AVD will appear as a window containing an emulated Android device environment. Figure 3-1, 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 3-1

3.2 Creating a New AVD

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

To create a new AVD, the first step is to launch the AVD Manager. This can be achieved from within the Eclipse environment using the *Window -> Android Virtual Device Manager* menu option. Alternatively, the tool may be launched from the command-line using the following command:

```
android avd
```

Once launched, the tool will appear as outlined in Figure 3-2. Assuming a new Android SDK installation, no AVDs will currently be listed:

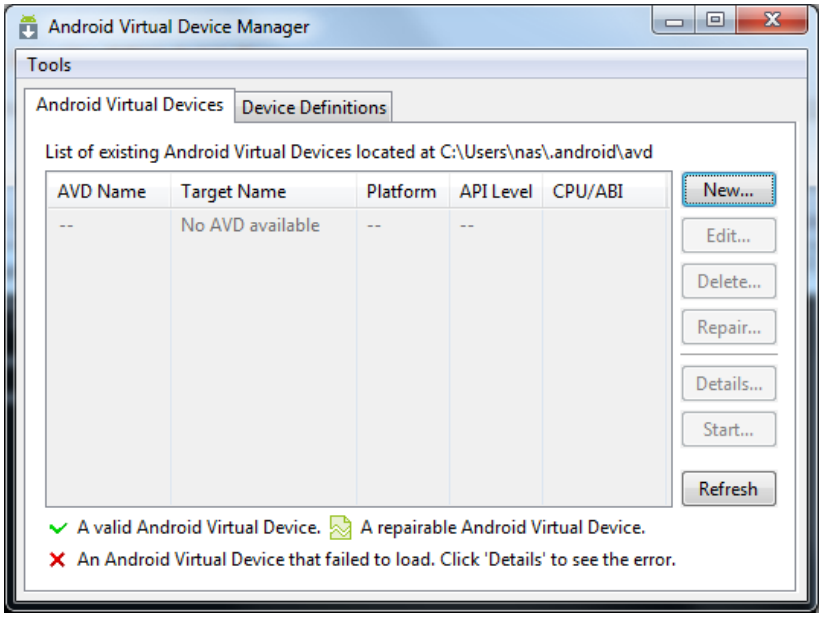

Figure 3-2

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 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 (7.27" 800 x 1280: tvhdpi).*
3. Set the *Target* menu to *Android 4.2.2 – API Level 17*.
4. Set the *CPU/ABI* menu to *ARM (armeabi-v7a)*.
5. 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.
6. 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* and *Display skin with hardware controls* options is optional. 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.

The skin with hardware controls option controls whether or not buttons appear as part of the emulator to simulate the hardware buttons present on the sides of the physical Android device.

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 3-3 illustrates the dialog with the appropriate settings implemented for a Nexus 7 emulator. Once the configuration settings are complete, click on the *OK* button.

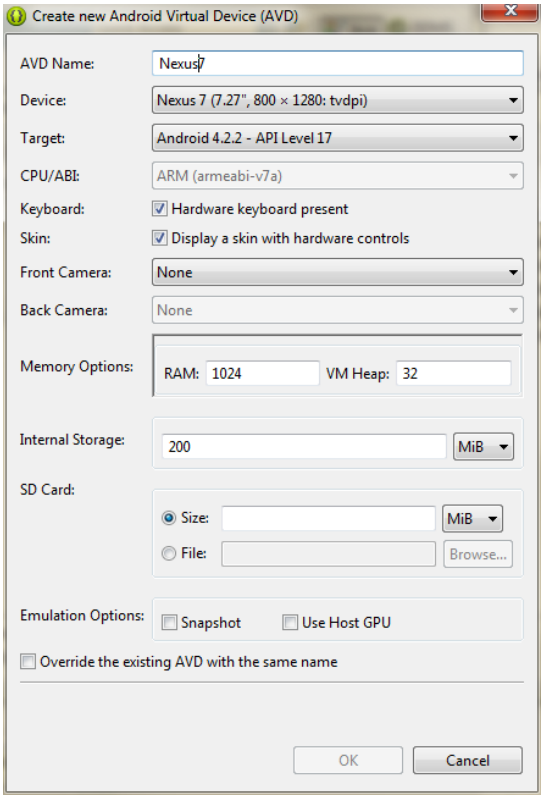

Figure 3-3

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.

3.3 Starting the Emulator

To perform a test run of the newly created AVD emulator, simply select the emulator from the Android Virtual Device 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 ADT system will detect that it is already running and attach to it when applications are launched, thereby saving considerable amounts of startup time.

Once fully loaded, the emulator will display either the standard Android lock screen.

3.4 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 *android* 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:

```
android 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-17"
     Name: Android 4.2.2
     Type: Platform
     API level: 17
     Revision: 2
     Skins: HVGA, QVGA, WQVGA400, WQVGA432, WSVGA, WVGA800 (default),
WVGA854, WXGA720, WXGA800, WXGA800-7in
     ABIs : armeabi-v7a
----------
id: 2 or "Google Inc.:Google APIs:17"
     Name: Google APIs
     Type: Add-On
     Vendor: Google Inc.
     Revision: 3
     Description: Android + Google APIs
     Based on Android 4.2.2 (API level 17)
     Libraries:
      * com.google.android.media.effects (effects.jar)
          Collection of video effects
      * com.android.future.usb.accessory (usb.jar)
          API for USB Accessories
      * com.google.android.maps (maps.jar)
          API for Google Maps
     Skins: WVGA854, WQVGA400, WSVGA, WXGA800-7in, WXGA720, HVGA, WQVGA432,
WVGA800 (default), QVGA, WXGA800
     ABIs : armeabi-v7a
----------
```

The syntax for AVD creation is as follows:

```
android create avd -n <name> -t <targetID> [-<option> <value>]
```

For example, to create a new AVD named *Nexus7* using the target id for the Android 4.2.2 API level 17 device (in this case id 1), the following command may be used:

```
android create avd -n Nexus7 -t 1
```

The android tool will create the new AVD to the specifications required for an Android 4.2.2 device. 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:

```
android list avd

C:\Users\nas>android list avd
Available Android Virtual Devices:
---------
    Name: KindleFireHD7
    Path: C:\Users\nas\.android\avd\KindleFireHD7.avd
  Target: Kindle Fire HD 7" (Amazon)
          Based on Android 4.0.3 (API level 15)
     ABI: armeabi-v7a
    Skin: 800x1280
---------
    Name: Nexus7
    Path: C:\Users\nas\.android\avd\Nexus7.avd
  Target: Android 4.2.2 (API level 17)
     ABI: armeabi-v7a
    Skin: 800x1280
---------
    Name: Nexus7Google
    Path: C:\Users\nas\.android\avd\Nexus7Google.avd
  Target: Google APIs (Google Inc.)
          Based on Android 4.2.2 (API level 17)
     ABI: armeabi-v7a
    Skin: 800x1280
```

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

```
android delete avd –name <avd name>
```

3.5 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
<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.

3.6 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 *android* tool's *move avd* argument. For example, to rename an AVD named Nexus7 to Nexus7B, the following command may be executed:

```
android move avd -n Nexus7 -r Nexus7B
```

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

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

For example, to move an AVD from its current file system location to /tmp/Nexus7Test:

```
android move avd -n Nexus7 -p /tmp/Nexus7Test
```

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

3.7 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.

Now that we have created and configured an AVD, the next step is to create a simple application and run it within the AVD.

Chapter 4

4. Creating an Example Android Application

The preceding chapters of this book have covered the steps necessary to configure an environment suitable for the development of Android applications. 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 a simple Android application, compile it and then run it within an Android Virtual Device (AVD) emulator.

4.1 Creating a New Android Project

The first step in the application development process is to create a new project within the Eclipse IDE. Begin, therefore, by launching Eclipse and accepting the default path to your workspace in the *Workspace Launcher* dialog as illustrated in Figure 4-1 (or choose another location if the default is unsuitable). Note that if you do not wish to be prompted for the location of the workspace each time Eclipse loads, simply select the *Use this as the default and do not ask again* option before clicking on *OK*.

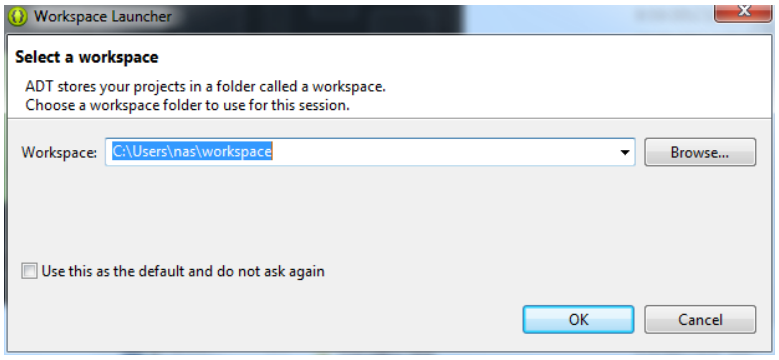

Figure 4-1

Once the workspace has been selected, the main Eclipse workbench window will appear ready for a new project to be created. To create the new project, select the *File -> New -> Android Application Project* menu option.

4.2 Defining the Project Name and SDK Settings

In the *New Android Project* window set both the *Application Name* and *Project Name* to *AndroidTest*.

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 *AndroidTest*, then the package name might be specified as:

```
com.mycompany.androidtest
```

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

```
com.example.androidtest
```

The next step is to specify some SDK settings. For the purposes of this example, the *Minimum Required SDK*, *Target SDK* and *Compile With* menus should all be set to *API 17: Android 4.2 (Jelly Bean)*. Once these settings have been configured, the dialog should match that shown in Figure 4-2:

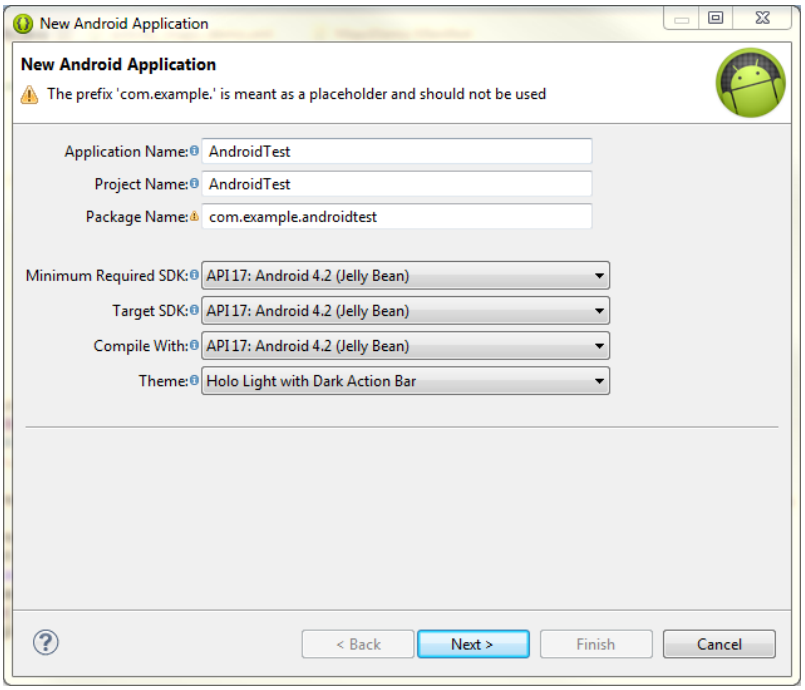

Figure 4-2

4.3 Project Configuration Settings

With the correct settings configured, click *Next >* to proceed to the *Configure Project* screen (Figure 4-3). Within this screen, a number of different configuration options are provided.

Make sure that the *Create Activity* and *Create customer launcher icon* options are selected. The former setting will ensure that the project is preconfigured with a template activity that will make the task of creating an example application easier. An activity is a single task that can be performed by the user within the context of an application and is typically analogous to a single user interface screen within an application. In asking for Eclipse to create an activity for us, therefore, the project will be primed with both a window onto which a user interface may be displayed and the code to ensure that the window appears when the application runs.