buzztouch for Android

A User's Guide

By GoNorthWest

June 2012

Table of Contents

Chapter 1

Welcome!

So, you've stumbled across **buzztouch**, the most amazing app development site on the entire Internet (probably in the entire universe as well!), and you're super stoked to get to work and send your ideas out to the world! Very cool! When I came across the site, I just wanted to get my app developed and out to the masses. But it turns out that the whole process is a bit more involved than just designing your app on buzztouch and then releasing it on the market. You need to download the source code, compile it, test it in a simulator and a real device (if you have one), sign it, and upload it to the appropriate market. This may seem like a lot of work, and it is, but it's not as hard as you might think. The purpose of this guide is to walk you step-by-step through the process of signing up for buzztouch, creating your app, setting up your Android development environment, and publishing your app for users to download. Hopefully this guide will help even the most inexperienced of users get up and running quickly and painlessly!

This book will be free to begin with. I do know that some people feel better if they can send something my way, so here are two methods you can use to do that. Like I said…nothing is expected, but everything is greatly appreciated!

PayPal Donation:

https://www.paypal.com/cgi-bin/webscr?cmd=_s-xclick&hosted_button_id=EC3HPWYK9QH96

Amazon.com Wish List:

http://amzn.com/w/1Y1ZG2R9UQCXC

Ready to start? Good! Let's go!

Chapter 2

Sign Up For buzztouch

It would be easy to assume that you've already signed up for buzztouch at this point, and perhaps you have, but it's entirely possible that you're reading this book to get an idea of what it's all about. Fair enough. If you haven't signed up for buzztouch yet, and you're ready to take the plunge, then head over to this link and complete your registration:

http://www.buzztouch.com/pages/register.php

Once you have created the registration form, a confirmation email will be sent to you with a link to a confirmation screen. Click that link, and you'll be confirmed and ready to go! Now, and in the future, head to the following link and sign in to your account:

http://www.buzztouch.com/pages/login.php

You're now signed up and ready to start developing your Android applications! But wait…I think it would be highly worth your time to spend some time at buzztouch U!

Time for buzztouch U!

Now that you've signed up for buzztouch, you absolutely should block out some time and go through the videos at buzztouch U. You can find them here:

http://www.buzztouch.com/account/btu.php

You should start with the **buzztouch Learning Path** so you fully understand what buzztouch is about, and how to operate within it. After that, it's your choice on either the iOS or Android path. I'm assuming since you're reading this book that you'll want to follow the **Android Learning Path**. In those 9 lessons, you will learn a ton about Android, Eclipse, and how programming works in general. You'll be amazed at the things you'll walk away with!

This book should be used in conjunction with the buzztouch U videos. It is NOT meant to replace the learning that can be found in those videos; rather, it is meant to help get you through some of the various problems that inevitably come up during the development process. You cannot go wrong by taking the time to engage in buzztouch U!

Chapter 3

Basic Hardware and Software Requirements

In order to develop apps for Android devices, there are some things that must be put into place first. buzztouch enables you to “design” your app, and then provides the source code for your project, but you then **have to compile, test, and submit your app** to the appropriate marketplace. In order to do that, your development environment must meet some minimum requirements.. So, here we go with a list of what you need to have in order to make your app dreams come true!

- ☐ **A Windows or Linux PC, or a Mac.** The awesome thing about the Android development environment is that it’s available on Windows, Linux or Mac. So, if you want to develop iOS and Android applications, getting a Mac will help you do both. If you are interested in just Android apps, then your current PC is probably up to the task.

- ☐ **Eclipse + Android SDK.** Eclipse is the Integrated Development Environment (IDE) for Android, and Google provides the Android Software Development Kit (SDK). You need both of these in order to compile the projects you get from buzztouch. Both have packages that run on Windows, Linux or Mac.

- ☐ **Membership in Android Market or Amazon.com Appstore for Android.** Technically speaking, this is not a requirement for distributing your apps to the world. All you need for that is a development certificate (which you can generate for free, unlike with Apple), and a way to get your apk file to somebody. But, if you want to use the conventional method for distributing your app, and increase your chances of making money from it, you’ll probably want to join one of these programs.

- ☐ **A device to test on.** OK…this is not an absolute requirement, but testing your app on more than just the Android emulator is very important. An emulator is just that…it emulates what a real device is like. If you don’t have a real device, find a friend that does and have them test it…it’s as simple as emailing the apk to them or posting it on the web. Or, post a message on the forum...lots of people are willing to help out.

- ☐ **Willingness to explore and learn.** Despite how accessible buzztouch has made app development, it’s not as easy as you might think, and Android apps have their fair share of issues in the development process. It’ll be immensely helpful to yourself, and to those who eventually respond to your questions in the forums, if you have spent some time trying to learn the required skills by utilizing the resources already out there. Please don’t expect us to lead you step-by-step through the process if you’re not willing to give it a go first. You’ll learn and understand more in the long run if you spend some time trying to work it out on your own first. It’s far more rewarding that way!

Helpful Resources

- ☐ Eclipse : http://www.eclipse.org/
- ☐ Android SDK : http://developer.android.com/sdk/index.html
- ☐ buzztouch Forum : http://www.buzztouch.com/forum/
- ☐ buzztouch Videos : http://www.buzztouch.com/videos/
- ☐ Setting up your Android Dev Environment : http://www.buzztouch.com/resources/Setting_Up_Your_Android_Development_Environment_v1.1.pdf
- ☐ Problem Reporting and Basic Debugging : http://www.buzztouch.com/resources/Problem_Reporting_and_Basic_Debug_v1.0.pdf

Chapter 4

Setting Up Your Android Development Environment (Mac OS X)

Setting up the Android development environment can be a bit…well…challenging if you don't have all the pieces in place before you start. The aim of this chapter is to visually walk you through the process of downloading what you need, installing it, and making the correct configurations. Once you do that, you're all set to actually work with the source code you download from buzztouch! Hopefully these instructions will make things just a bit easier for you!

The basic process you are about to undertake is the following:

1. Download the required software
2. Unzip and move the Android SDK to the desired location
3. Unzip and move the Eclipse IDE to the desired location
4. Install the ADT Plugin for Eclipse
5. Configure the Android SDK for Development Tools and Google API
6. Configure the Android Virtual Device (AVD)
7. Celebrate because you're done and ready to work with your app!

Note: This tutorial assumes a **fresh install** of all software components. If you've got a previous install, consider deleting all the folders and starting fresh. But remember to copy over any extra files or keys you may have created!

Let's get started!

1. **Download required software**. In order to make this all work, there are several pieces of software you have to download and install:

    a. **Eclipse**: I chose to download Eclipse Classic 3.7.2, which was the most recent version available at the time of this writing. I downloaded the 32-bit version for Mac OS X, but there is also a 64-bit version if you are running 64-bit OS X. Here are the links I used to get this package.

       http://www.eclipse.org/downloads/?osType=macosx

       http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.7.2-201202080800/eclipse-SDK-3.7.2-macosx-cocoa.tar.gz http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.7.2-201202080800/eclipse-SDK-3.7.2-macosx-cocoa-x86_64.tar.gz

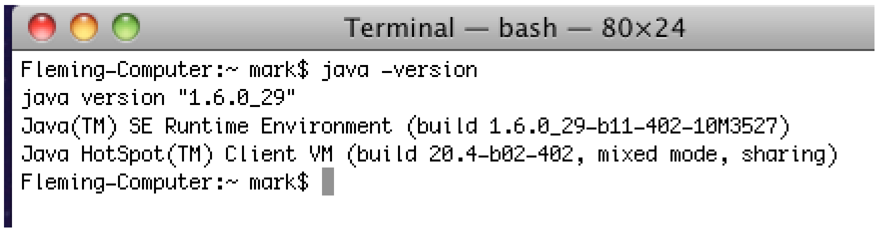
b. **Android Development SDK:** This is the software development kit (SDK), much like the iOS SDK for iPhones/iPads, that must be used within Eclipse for development. Here are the links I used for this portion.

http://developer.android.com/sdk/index.html#installingplugin

http://dl.google.com/android/android-sdk_r17-macosx.zip

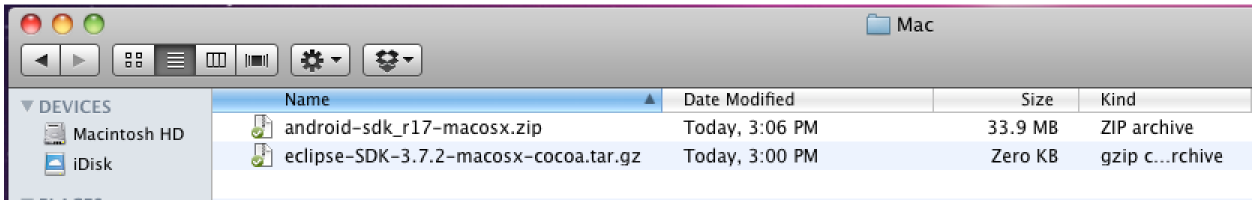
http://developer.android.com/sdk/eclipse-adt.html#installing

c. **Java JDK:** Windows has the requirement of a JDK, but it appears that the Java that comes with Mac OS X is enough for our needs here. So, there is nothing you need to download. If you want to check what version of Java is running on your computer, you can open a console window and type 'java –version' and you'll get something like the following:



```
Fleming-Computer:~ mark$ java -version
java version "1.6.0_29"
Java(TM) SE Runtime Environment (build 1.6.0_29-b11-402-10M3527)
Java HotSpot(TM) Client VM (build 20.4-b02-402, mixed mode, sharing)
Fleming-Computer:~ mark$
```

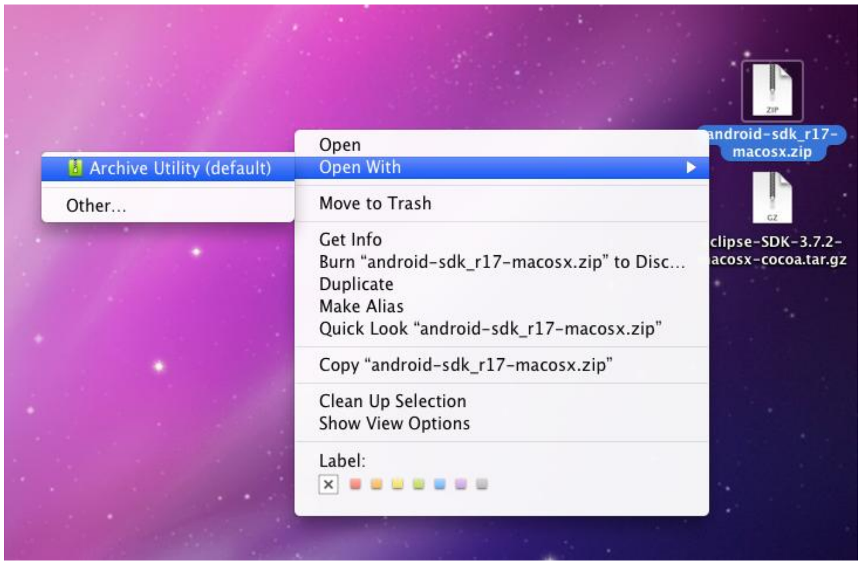
d. Once you get all the software downloaded, you should have a file listing that looks something like this :



Name	Date Modified	Size	Kind
android-sdk_r17-macosx.zip	Today, 3:06 PM	33.9 MB	ZIP archive
eclipse-SDK-3.7.2-macosx-cocoa.tar.gz	Today, 3:00 PM	Zero KB	gzip c...rchive

2. **Install the Android SDK (android-sdk_r17-macosx.zip).** To install the Android SDK in Mac OS X, all you need to do is unzip the package that you downloaded, and place it wherever you want to on your system. I chose to put the folder in the Applications folder, but I don't think it much matters. You may have to adjust a few things that come later depending on where you place the folder.

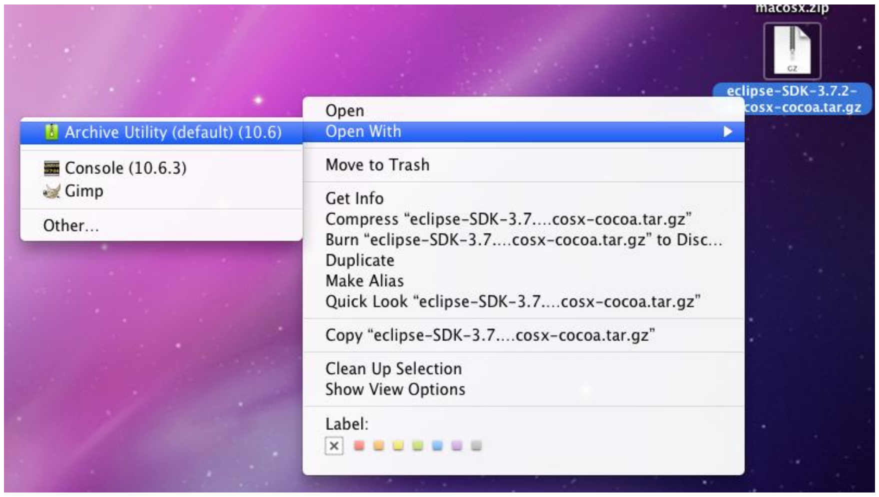
   Begin by right-clicking on the package you downloaded (I put mine on the Desktop), and using the Archive Utility to extract the files.


android-sdk_r17-macosx.zip
eclipse-SDK-3.7.2-macosx-cocoa.tar.gz
Archive Utility (default)
Other...
Open
Open With
Move to Trash
Get Info
Burn "android-sdk_r17-macosx.zip" to Disc...
Duplicate
Make Alias
Quick Look "android-sdk_r17-macosx.zip"
Copy "android-sdk_r17-macosx.zip"
Clean Up Selection
Show View Options
Label:

After you have extracted the files, you'll see a newly created folder called **android-sdk-macosx**. Drag that folder to wherever you want. Like I mentioned, I put it in the Applications folder.

3. **Install Eclipse IDE (eclipse-SDK-3.7.2-macosx-cocoa.tar.gz).** To install the Eclipse IDE in Mac OS X, you again simply need to extract the files from the zipped tar package, and then move them to wherever you want Eclipse to reside. And, again, I chose that location to be the Applications folder.

   Begin by right-clicking on the package you downloaded and using the Archive Utility to extract the files.

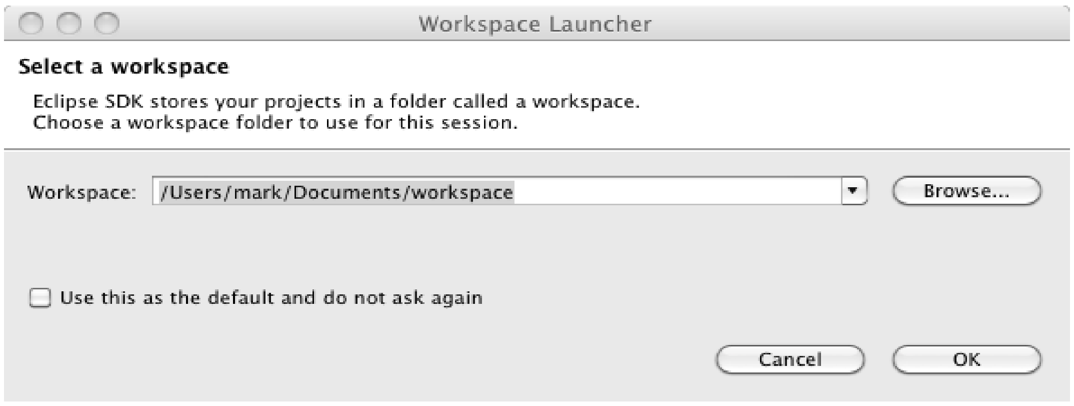

macosx.zip
eclipse-SDK-3.7.2-macosx-cocoa.tar.gz
Archive Utility (default) (10.6)
Console (10.6.3)
Gimp
Other...
Open
Open With
Move to Trash
Get Info
Compress "eclipse-SDK-3.7....cosx-cocoa.tar.gz"
Burn "eclipse-SDK-3.7....cosx-cocoa.tar.gz" to Disc...
Duplicate
Make Alias
Quick Look "eclipse-SDK-3.7....cosx-cocoa.tar.gz"
Copy "eclipse-SDK-3.7....cosx-cocoa.tar.gz"
Clean Up Selection
Show View Options
Label:

After you have extracted the files, you'll see a newly created folder called **eclipse**. Drag that folder to wherever you want. I put mine in the Applications folder.

You're done installing Eclipse and the Android SDK. Now it's time to install the **ADT Plugin for Eclipse.**

4. **Install the ADT Plugin for Eclipse.** This part of the procedure installs the Android Development Tools (ADT) plugin for Eclipse. It's what ties together Eclipse and the Android SDK. You install it via Eclipse, using the Help…Install New Software menu of Eclipse.

   First thing is to fire up Eclipse. Double-click on the Eclipse executable in the Applications/Eclipse folder (or wherever you put it), and you'll see the Eclipse splash screen. The first thing you'll have to decide is where to configure your workspace. Unless you have a reason to change from the default, go with what it suggests.


Workspace Launcher

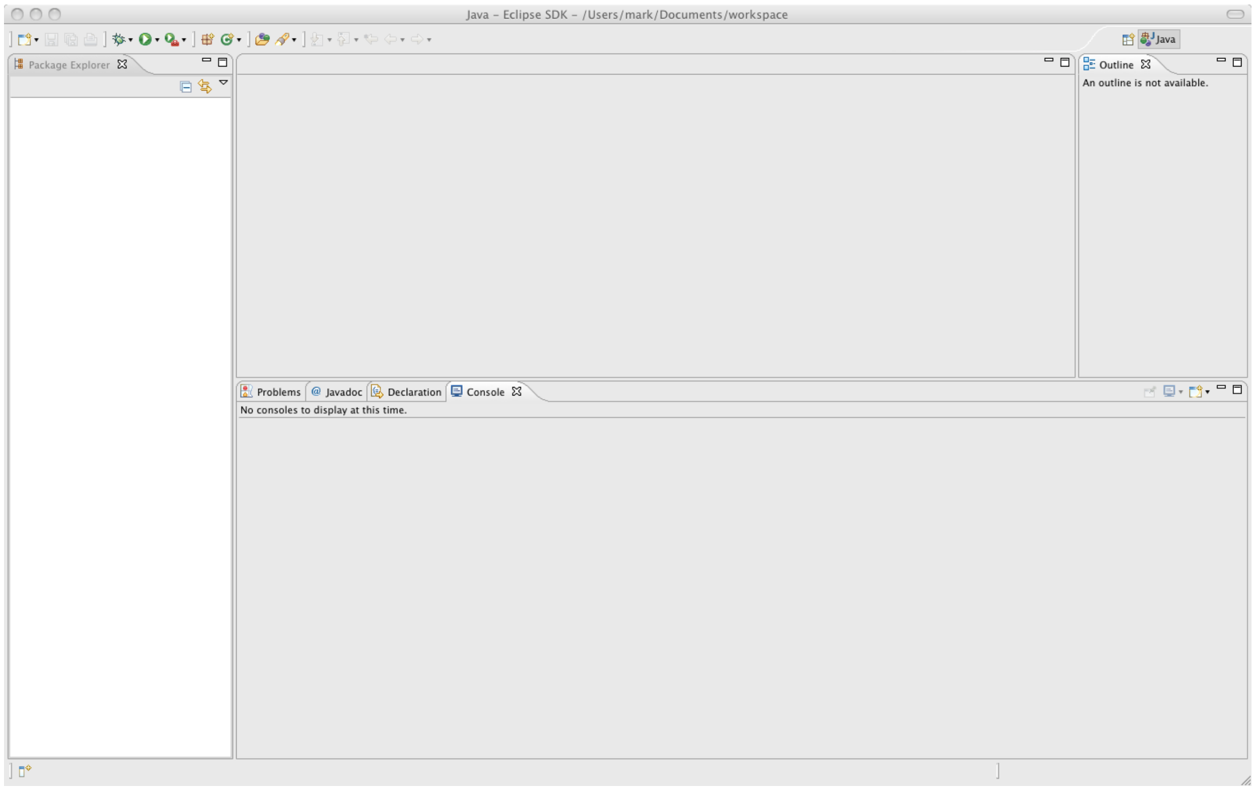
Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /Users/mark/Documents/workspace

☐ Use this as the default and do not ask again

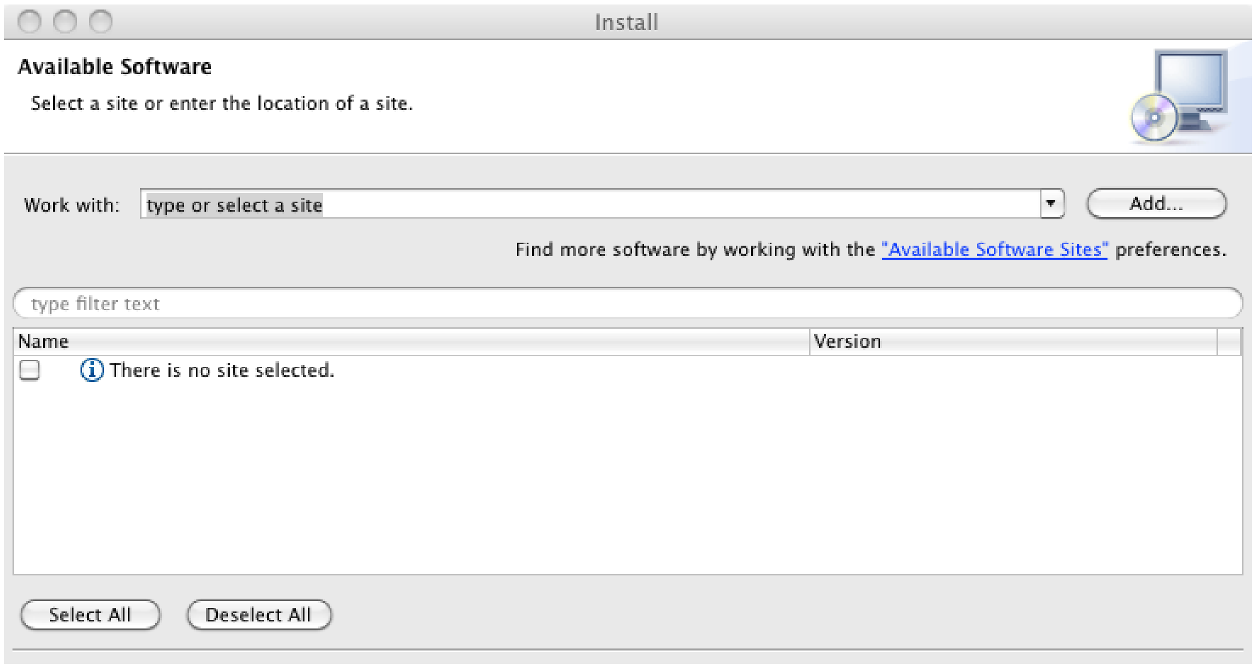
   One you have that set, and hit OK, the main Eclipse screen should come up. Do a happy dance now, because this is proof positive that Eclipse is installed on your system!



An outline is not available.

No consoles to display at this time.

Now you want to go to Help…Install New Software, and configure the screen to hit the correct software repository. Here is the initial screen you'll get.



Available Software

Select a site or enter the location of a site.

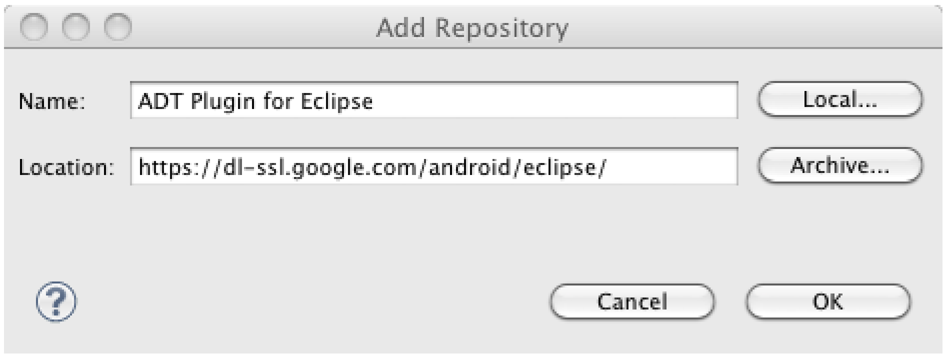
Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☐ There is no site selected.	

I cut off the bottom part of the actual screen you'll get to save space. The important stuff is at the top of the screen anyway. See the **Add** button on the right-hand side near the top? Click that puppy, and here's what you'll get.

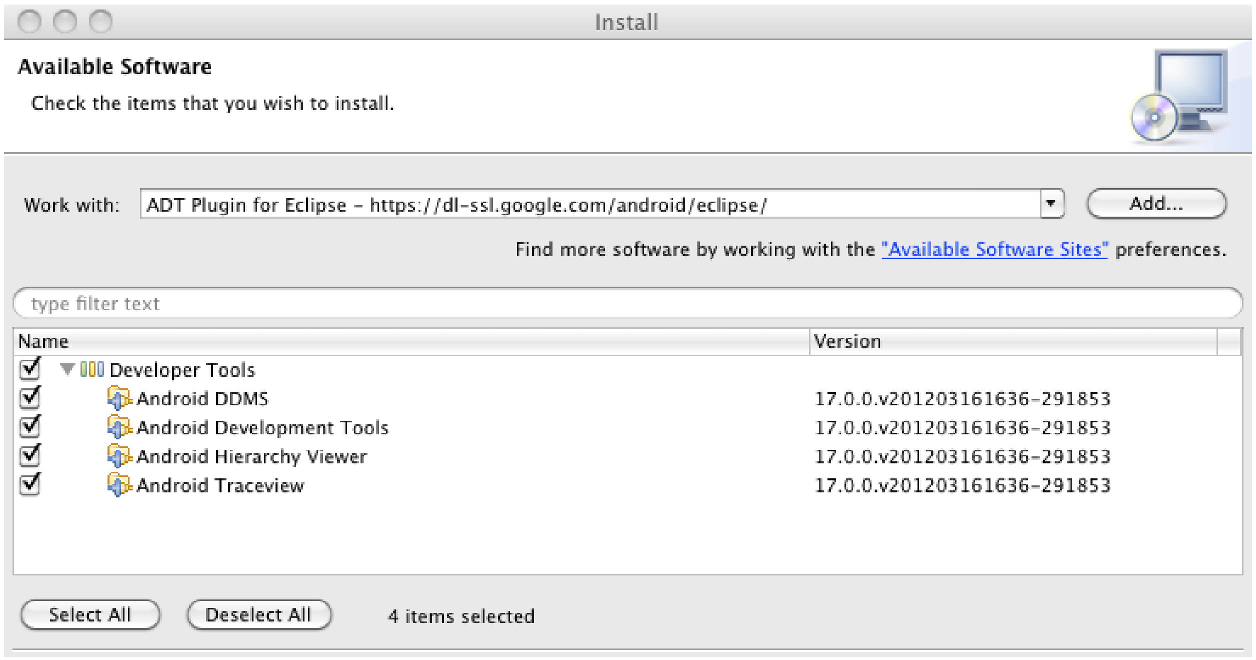


Add Repository

Name: ADT Plugin for Eclipse

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

OK, that's not totally what you'll get. You'll need to type in both the Name (**ADT Plugin for Eclipse**) and the Location (**https://dl-ssl.google.com/android/eclipse/**). Now you can hit OK. After a few seconds it'll connect to the Google site, and then you'll see this screen (again, cropped for space).



Install

Available Software

Check the items that you wish to install.

Work with: ADT Plugin for Eclipse - https://dl-ssl.google.com/android/eclipse/

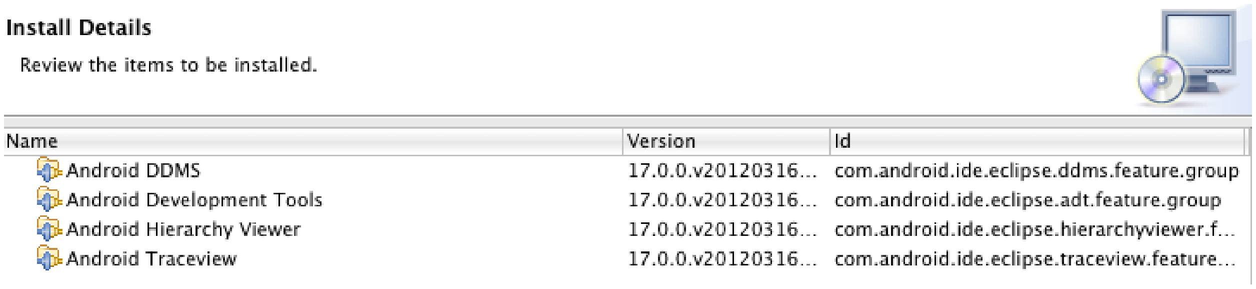
Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
✓ Developer Tools	
✓ Android DDMS	17.0.0.v201203161636-291853
✓ Android Development Tools	17.0.0.v201203161636-291853
✓ Android Hierarchy Viewer	17.0.0.v201203161636-291853
✓ Android Traceview	17.0.0.v201203161636-291853

4 items selected

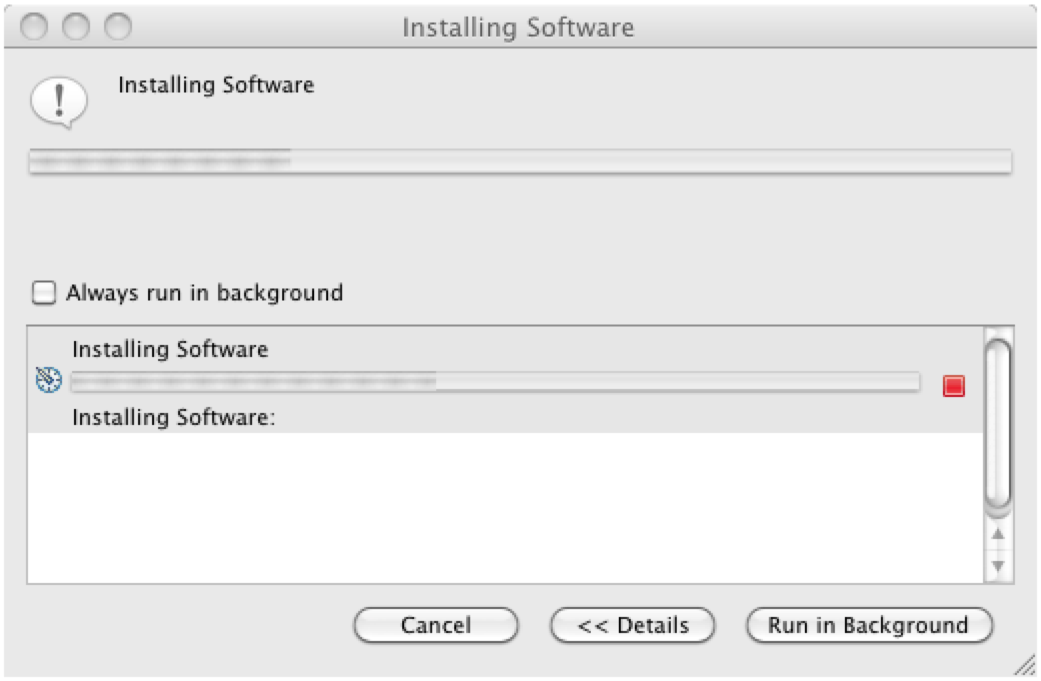
Go ahead and hit **Select All** as I've done in this example. To be honest, I don't know if you absolutely need everything here, but I figured better safe than sorry! Hit Next to review the **Install Details**.


Install Details

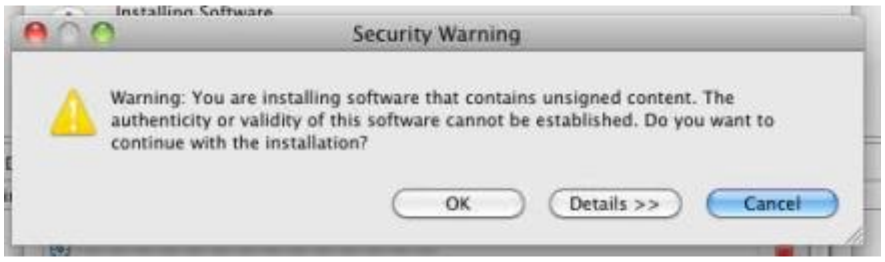
Review the items to be installed.

Name	Version	Id
Android DDMS	17.0.0.v20120316...	com.android.ide.eclipse.ddms.feature.group
Android Development Tools	17.0.0.v20120316...	com.android.ide.eclipse.adt.feature.group
Android Hierarchy Viewer	17.0.0.v20120316...	com.android.ide.eclipse.hierarchyviewer.f...
Android Traceview	17.0.0.v20120316...	com.android.ide.eclipse.traceview.feature...

Hit Next again, and select "**I accept the terms of the license agreements.**" That should enable the Finish button, which you should now hit. The following screen will pop up, which indicates you are now installing the software!

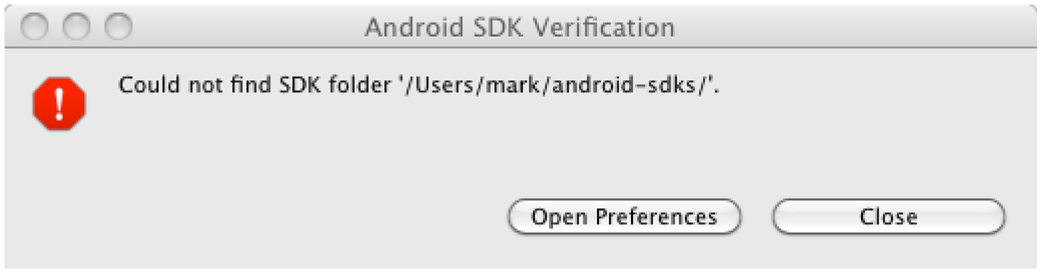


In the course of the install, you may see the following security warning. Just go ahead and hit **OK** and things will proceed.



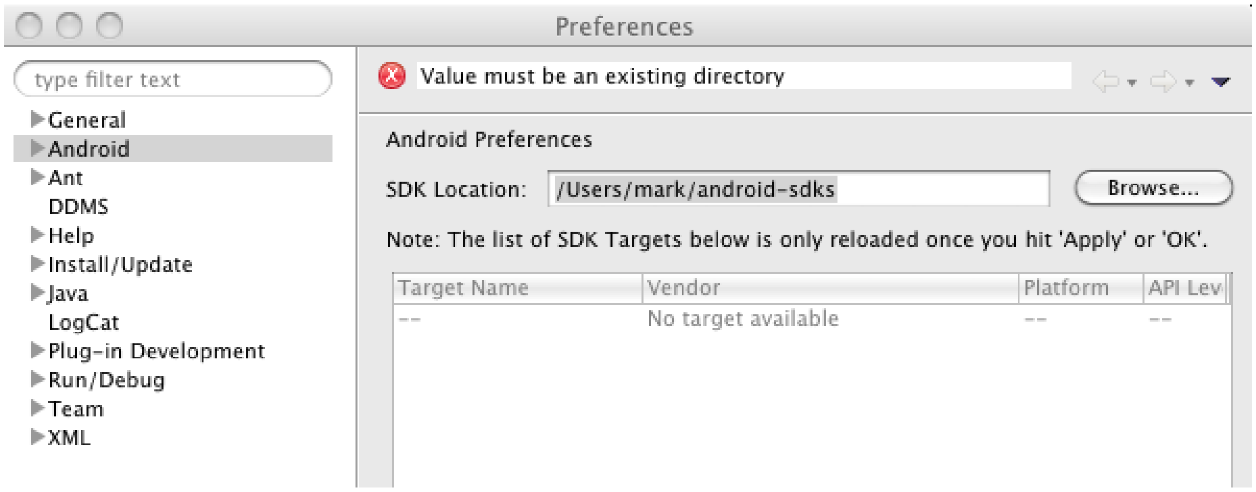
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

At this point, you may get a series of errors, and one of them may look like this.


Android SDK Verification

Could not find SDK folder '/Users/mark/android-sdks/'.

Click on **Open Preferences**, and see this screen (cropped for space).


Preferences

Value must be an existing directory

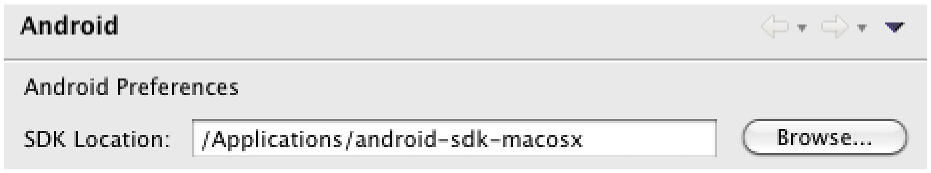
Android Preferences

SDK Location: /Users/mark/android-sdks

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API Lev
--	No target available	--	--

Click on **Browse** and navigate to where you put the Android SDK folder. In my example, it'll look like this (cropped!).


Android

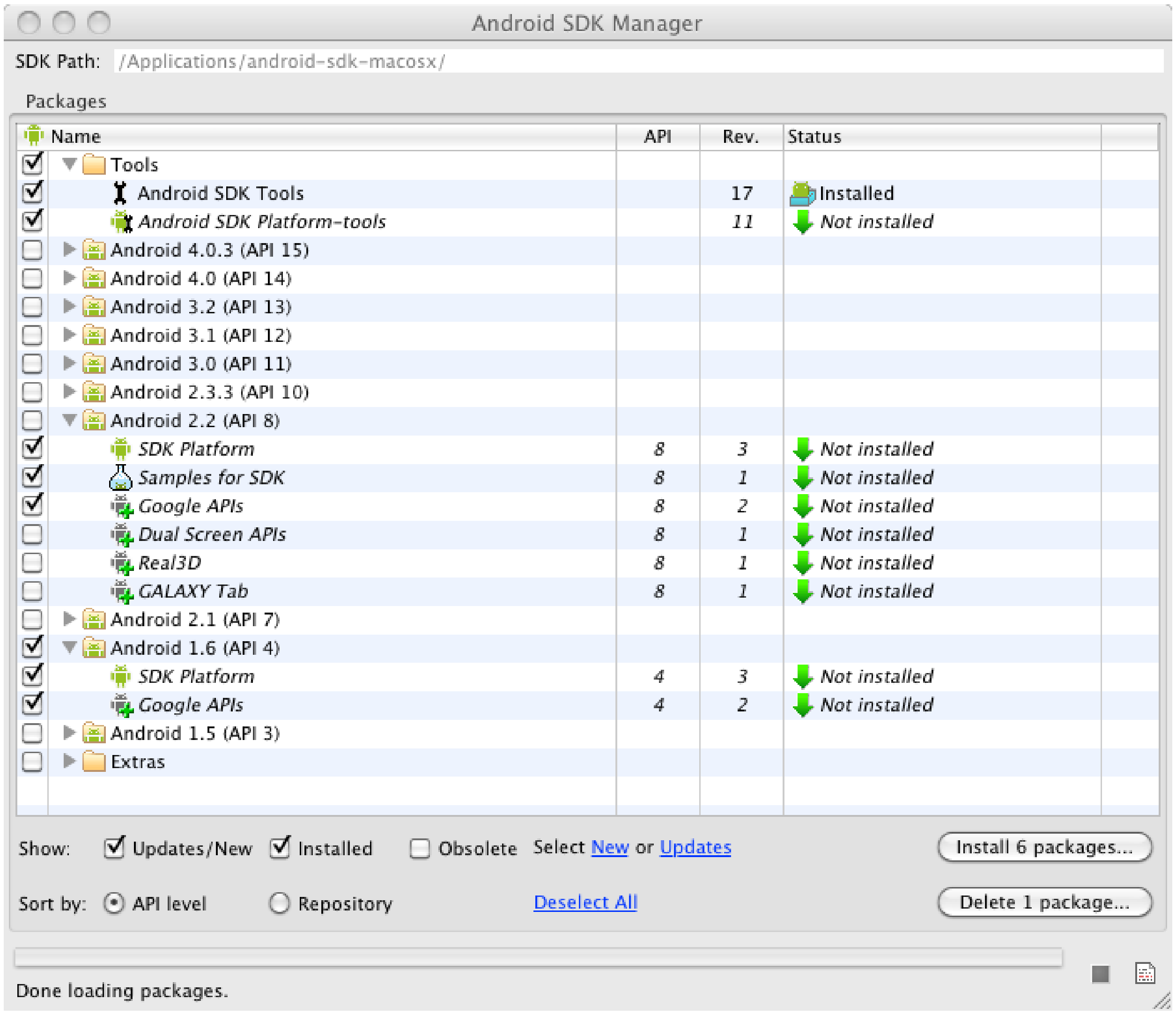
Android Preferences

SDK Location: /Applications/android-sdk-macosx

Click **Apply**, then **OK**, and you're all done! Now it's time to configure the Android SDK for Development Tools and the Google APIs. You're almost done!

5. **Configure the Android SDK for Development Tools and Google API.** This is where you configure the Android SDK Manager to download the tools you will need to create buzztouch apps. This is essentially downloading the Google API's Platform 2.2 and API level 8. Keep in mind a common mistake, as pointed out in the Readme file provided by buzztouch with your Android source code : "A common mistake is to assume Android 2.2 API's and Google 2.2 API's are the same, they are not." This is important to keep in mind…if you grab the wrong ones, you won't be able to get your program to work.

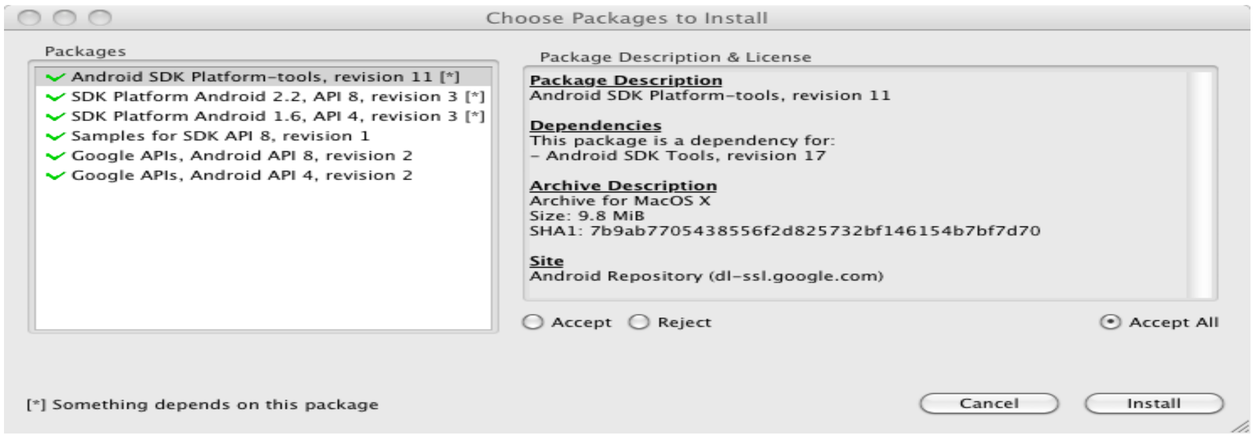
   Start by bringing up the **Android SDK Manager** in Eclipse. Do this by starting Eclipse, then going to **Window…Android SDK Manager**. The following screen will appear, and your goal is to make it look like the image you see here. Select what I selected, and you'll be good for both buzztouch v1.4, v1.5 and v2.0 apps.



	Name	API	Rev.	Status
✓	Tools			
✓	Android SDK Tools		17	Installed
✓	*Android SDK Platform-tools*		*11*	*Not installed*
	Android 4.0.3 (API 15)			
	Android 4.0 (API 14)			
	Android 3.2 (API 13)			
	Android 3.1 (API 12)			
	Android 3.0 (API 11)			
	Android 2.3.3 (API 10)			
	Android 2.2 (API 8)			
✓	*SDK Platform*	*8*	*3*	*Not installed*
✓	*Samples for SDK*	*8*	*1*	*Not installed*
✓	*Google APIs*	*8*	*2*	*Not installed*
	Dual Screen APIs	*8*	*1*	*Not installed*
	Real3D	*8*	*1*	*Not installed*
	GALAXY Tab	*8*	*1*	*Not installed*
	Android 2.1 (API 7)			
✓	Android 1.6 (API 4)			
✓	*SDK Platform*	*4*	*3*	*Not installed*
✓	*Google APIs*	*4*	*2*	*Not installed*
	Android 1.5 (API 3)			
	Extras			

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates
Install 6 packages...
Sort by: ● API level ○ Repository
Deselect All
Delete 1 package...
Done loading packages.

Now click **Install 6 packages** (or whatever number you end up having), and you'll get this screen.



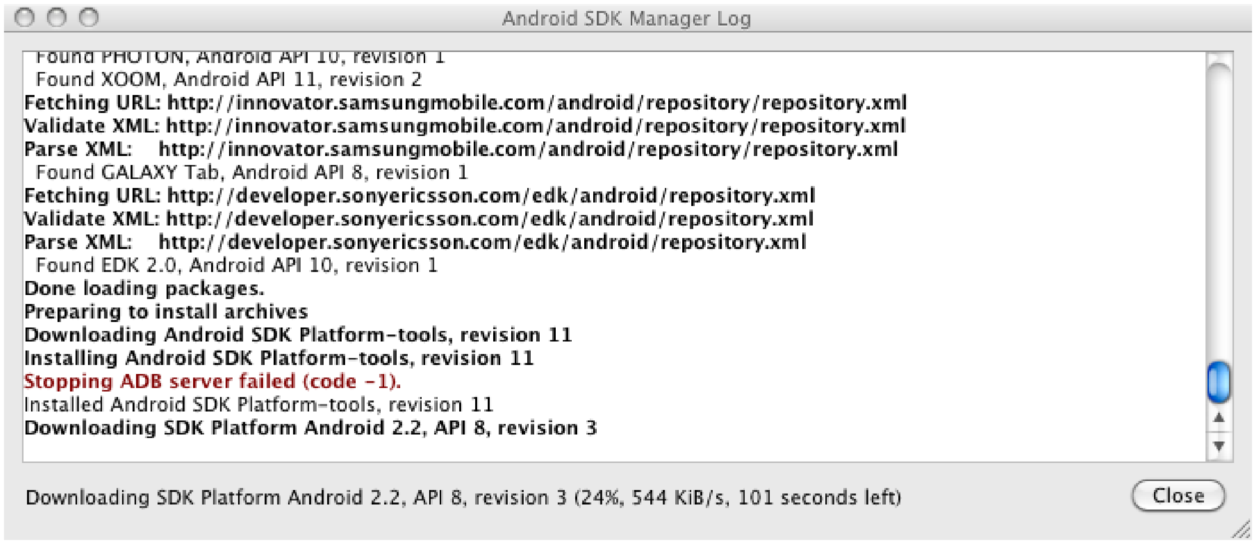
```
✓ Android SDK Platform-tools, revision 11 [*]
✓ SDK Platform Android 2.2, API 8, revision 3 [*]
✓ SDK Platform Android 1.6, API 4, revision 3 [*]
✓ Samples for SDK API 8, revision 1
✓ Google APIs, Android API 8, revision 2
✓ Google APIs, Android API 4, revision 2
```

```
Package Description
Android SDK Platform-tools, revision 11

Dependencies
This package is a dependency for:
- Android SDK Tools, revision 17

Archive Description
Archive for MacOS X
Size: 9.8 MiB
SHA1: 7b9ab7705438556f2d825732bf146154b7bf7d70

Site
Android Repository (dl-ssl.google.com)
```

Accept Reject [✓ Accept All]

[*] Something depends on this package

Click on the **Accept All** radio button, which should turn all the checkmarks in the packages list to green. Now hit **Install.**



```
  Found PHOTON, Android API 10, revision 1
  Found XOOM, Android API 11, revision 2
Fetching URL: http://innovator.samsungmobile.com/android/repository/repository.xml
Validate XML: http://innovator.samsungmobile.com/android/repository/repository.xml
Parse XML:    http://innovator.samsungmobile.com/android/repository/repository.xml
  Found GALAXY Tab, Android API 8, revision 1
Fetching URL: http://developer.sonyericsson.com/edk/android/repository.xml
Validate XML: http://developer.sonyericsson.com/edk/android/repository.xml
Parse XML:    http://developer.sonyericsson.com/edk/android/repository.xml
  Found EDK 2.0, Android API 10, revision 1
Done loading packages.
Preparing to install archives
Downloading Android SDK Platform-tools, revision 11
Installing Android SDK Platform-tools, revision 11
Stopping ADB server failed (code -1).
Installed Android SDK Platform-tools, revision 11
Downloading SDK Platform Android 2.2, API 8, revision 3
```

Downloading SDK Platform Android 2.2, API 8, revision 3 (24%, 544 KiB/s, 101 seconds left)

Watch the log until it says you are done and you can close. Then hit **Close.** This part of the process may take some time depending on the speed of your Internet connection, so feel free to grab a coffee or something to pass the time with.

Once the install is done, and it says you can hit Close, you've successfully installed the Android SDK, Eclipse, and the ADT Plugin for Eclipse! You are basically done, unless you want to create an Android Virtual Device (AVD), or emulator, as it's often called. If you're ready for that, head on over to **Chapter 6: Configuring Your Android Virtual Device (AVD).**

Congratulations on making it through one of the more difficult parts of developing for Android! Hopefully this chapter has made it as painless a process as possible.

Chapter 5

Setting Up Your Android Development Environment (Windows)

Like I said in the previous chapter, setting up the Android development environment can be a bit…well…challenging if you don't have all the pieces in place before you start. The aim of this chaper is to visually walk you through the process of downloading what you need, installing it, and making the correct configurations. Once you do that, you're all set to actually work with the source code you download from buzztouch! Hopefully these instructions will make things just a bit easier for you!

The basic process you are about to undertake is the following:

1. Download the required software
2. Use the installer package to install Java Developers Kit (JDK)
3. Use the installer package to install the Android SDK
4. Configure the Android SDK for Development Tools and Google API
5. Unzip and move the Eclipse IDE to the desired location
6. Install the ADT Plugin for Eclipse
7. Configure the Android Virtual Device (AVD)
8. Celebrate because you're done and ready to work with your app!

Note: This tutorial assumes a **fresh install** of all software components. If you've got a previous install, consider deleting all the folders and starting fresh. But remember to copy over any extra files or keys you may have created!

Let's get started!

1. **Download required software**. In order to make this all work, there are several pieces of software you have to download and install:

    a. **Java JDK:** Eclipse requires a Java Development Kit (JDK). You have to use the Java Developers Kit here…not just the Java Runtime Engine (JRE), which most of you already have installed. This is a very crucial distinction…without it, things will not work. Here are the links I used for this download, and I grabbed Java SE Development Kit 7u4 which was the latest version at the time of this writing.

    http://download.oracle.com/otn-pub/java/jdk/7u4-b22/jdk-7u4-windows-i586.exe

    http://download.oracle.com/otn-pub/java/jdk/7u4-b22/jdk-7u4-windows-x64.exe

b. **Eclipse**: I chose to download Eclipse Classic 3.7.2, which was the most recent version available at the time of this writing. I downloaded the 32-bit version for Windows, but there is also a 64-bit version if you are running 64-bit Windows. Here are the links I used to get this package.

   http://www.eclipse.org/downloads/?osType=win32

   http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.7.2-201202080800/eclipse-SDK-3.7.2-win32.zip

   http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.7.2-201202080800/eclipse-SDK-3.7.2-win32-x86_64.zip

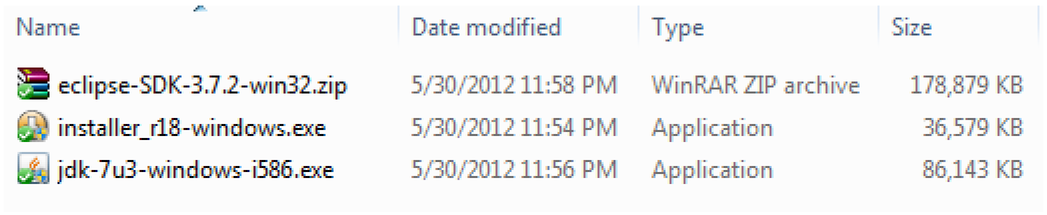
c. **Android Development SDK:** This is the software development kit (SDK), much like the iOS SDK for iPhones/iPads, that must be used within Eclipse for development. Here are the links I used for this portion.

   http://developer.android.com/sdk/index.html

   http://dl.google.com/android/installer_r18-windows.exe

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

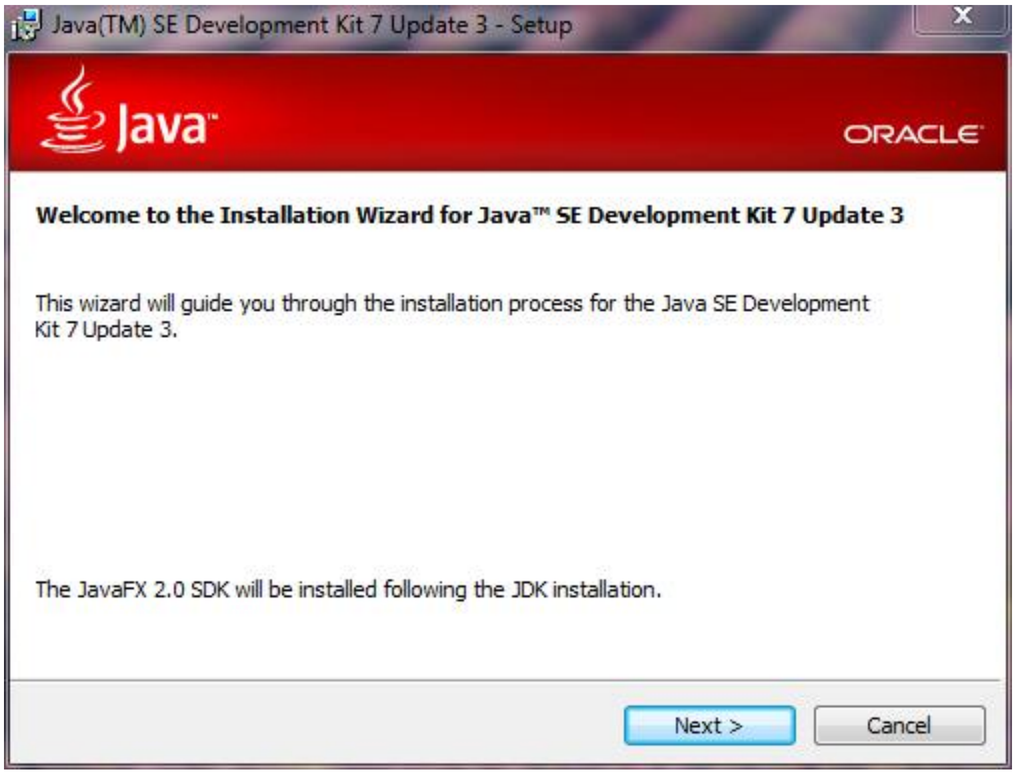
Once you get all the software downloaded, you should have a file listing that looks something like this:



Name	Date modified	Type	Size
eclipse-SDK-3.7.2-win32.zip	5/30/2012 11:58 PM	WinRAR ZIP archive	178,879 KB
installer_r18-windows.exe	5/30/2012 11:54 PM	Application	36,579 KB
jdk-7u3-windows-i586.exe	5/30/2012 11:56 PM	Application	86,143 KB

2. Use the installer package to install Java Developers Kit (JDK)

Double-click on the JDK install package you downloaded from above, and follow the prompts. This MUST be installed before you can install the Android SDK…without it, SDK installation will fail. Your install screens should look something like the following:

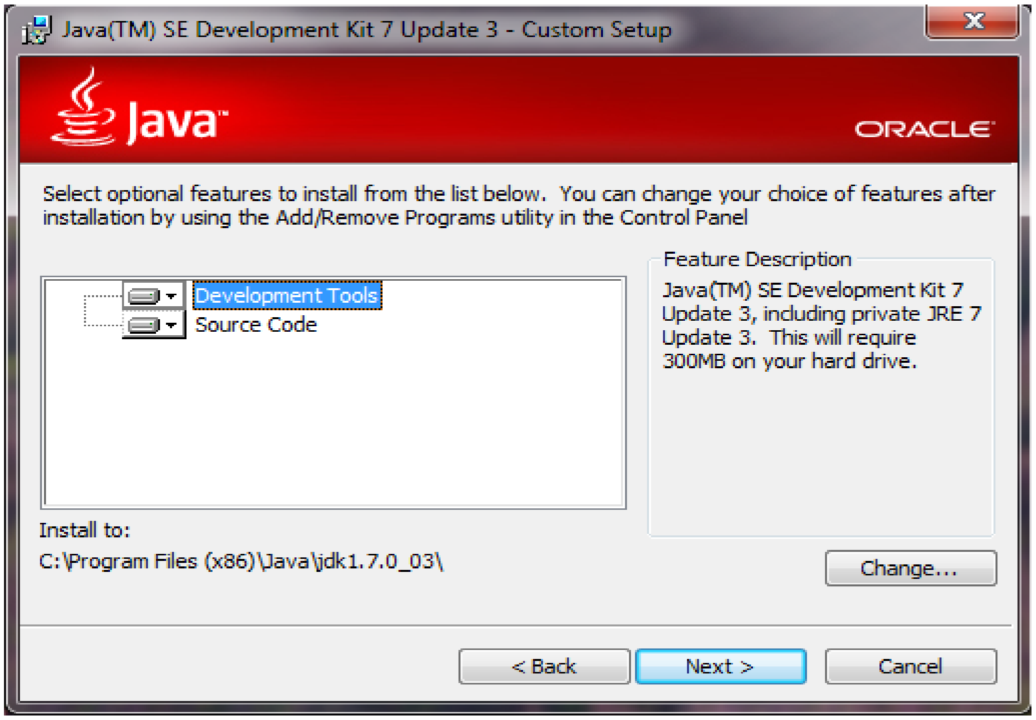


Welcome to the Installation Wizard for Java™ SE Development Kit 7 Update 3

This wizard will guide you through the installation process for the Java SE Development Kit 7 Update 3.

The JavaFX 2.0 SDK will be installed following the JDK installation.

Notice it says the JavaFX SDK will be installed following the JDK installation. You don't need that for Android development, so when the time comes for that, you can hit cancel with no problems.



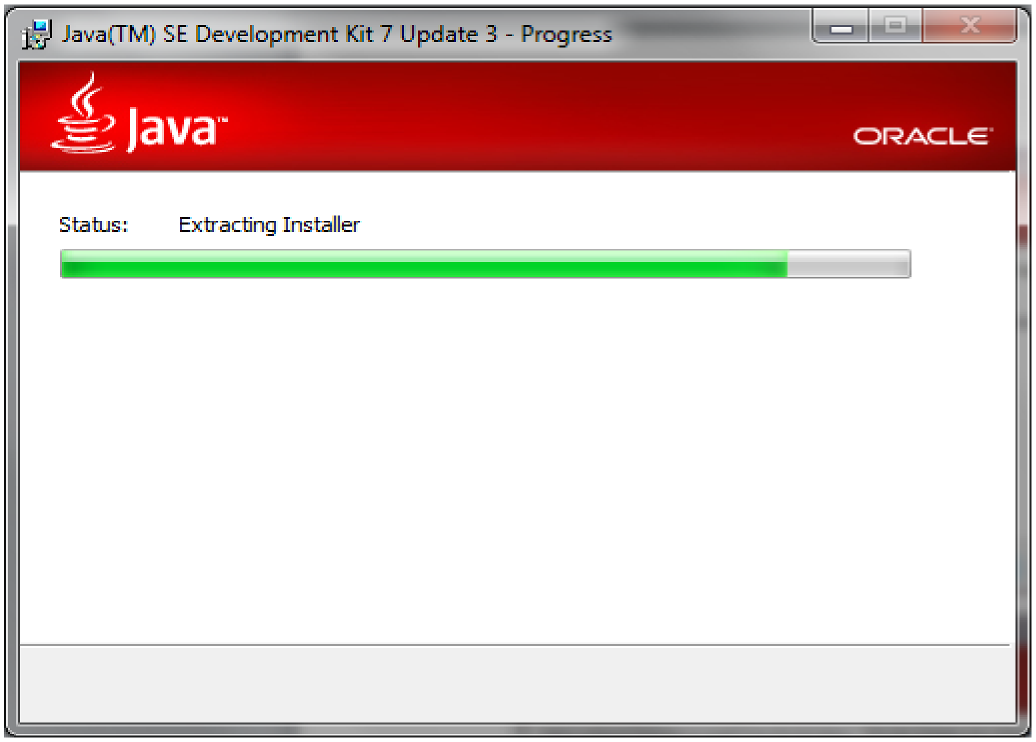
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

Development Tools
Source Code

Feature Description
Java(TM) SE Development Kit 7 Update 3, including private JRE 7 Update 3. This will require 300MB on your hard drive.

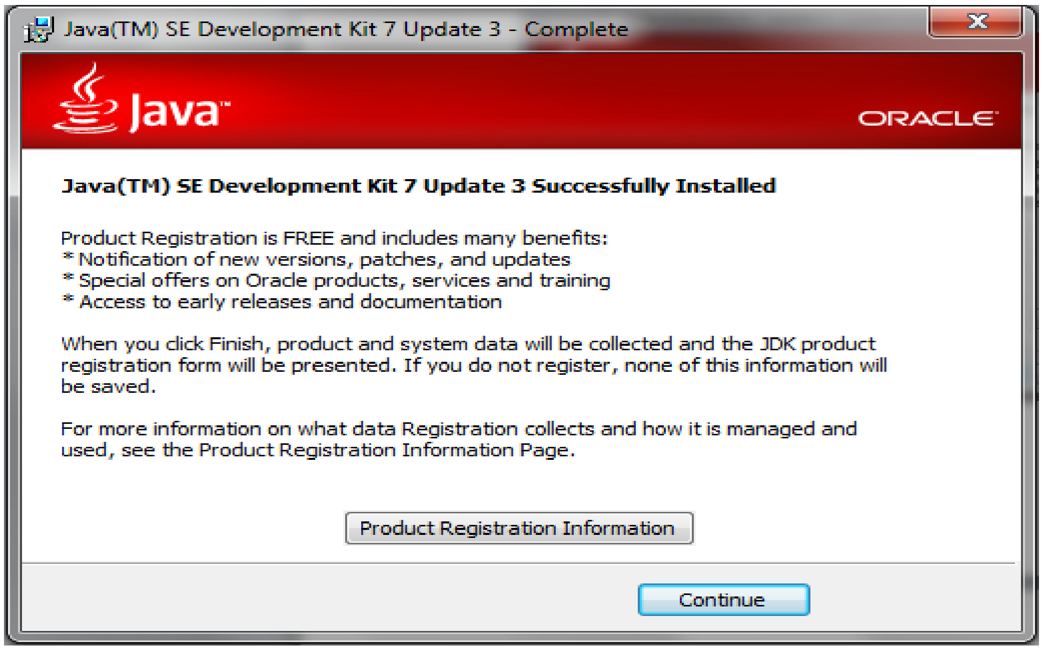
Install to:
C:\Program Files (x86)\Java\jdk1.7.0_03\

The screen above gives you the option on where to install the JDK (if you have a preference), and what components you want. Just go with the defaults unless you have a reason to change, and are comfortable making that selection.



Status: Extracting Installer

Kick back and watch the progress meter…it'll go fairly quickly. Once its done, you'll see the screen below, and you can hit continue to finish up!



Java(TM) SE Development Kit 7 Update 3 Successfully Installed

Product Registration is FREE and includes many benefits:
* Notification of new versions, patches, and updates
* Special offers on Oracle products, services and training
* Access to early releases and documentation

When you click Finish, product and system data will be collected and the JDK product registration form will be presented. If you do not register, none of this information will be saved.

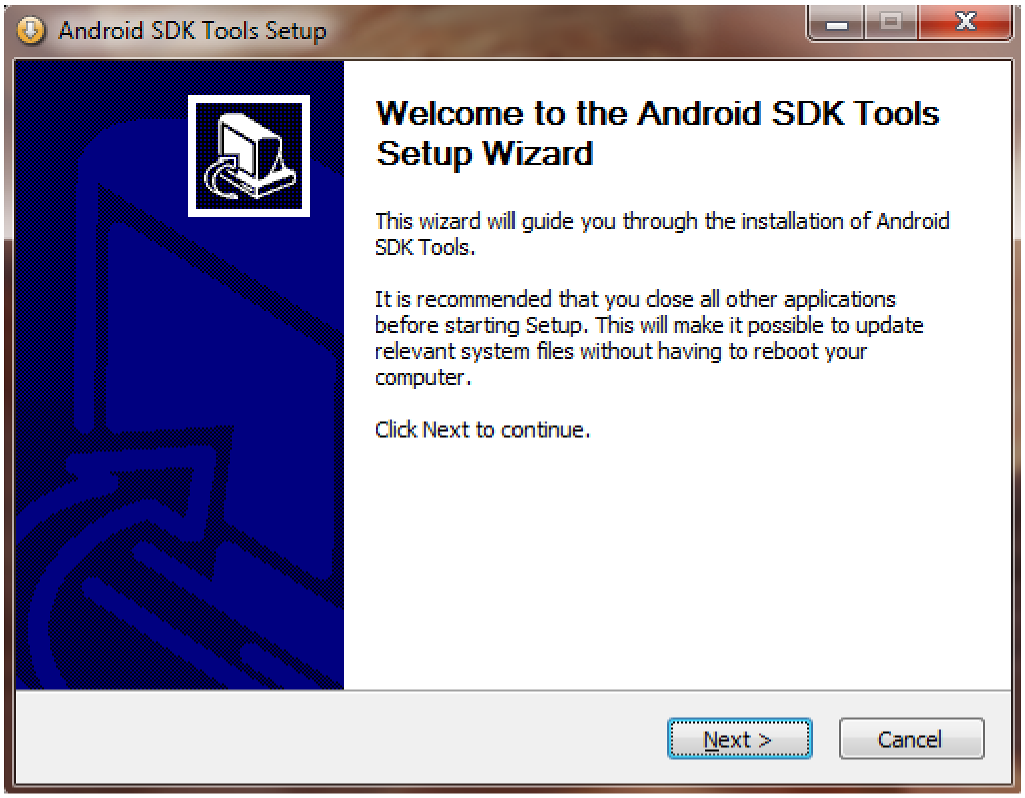
For more information on what data Registration collects and how it is managed and used, see the Product Registration Information Page.

At this point it'll ask you about installing JavaFX SDK. You don't need it, so you can hit Cancel safely, and you're done. The first piece of the puzzle is now in place!

3. **Use the installer package to install the Android SDK**

Now it's time to install the Android Software Development Kit (SDK). You might be thinking to yourself that you need to install Eclipse first, but that's not true, and we can save that for later on. In fact, it's the Android Development Tools (ADT) plugin for Eclipse that actually ties the Android stuff to Eclipse, so we have plenty of time to get things in order.

Double-click on the Android SDK install package you downloaded from above, and follow the prompts. Your install screens should look something like the following:



Welcome to the Android SDK Tools Setup Wizard

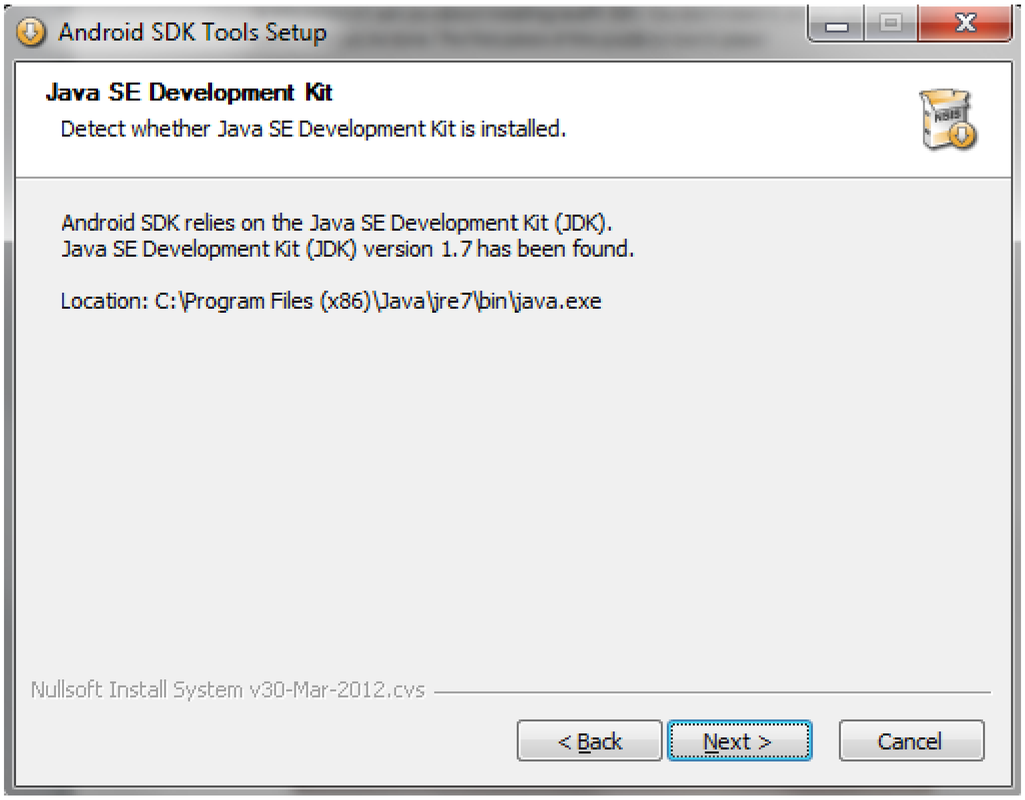
This wizard will guide you through the installation of Android SDK Tools.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Hit Next, and you'll see the screen below. Sometimes, for reasons unknown, it will fail to find a JDK, even though there is one installed. If that happens to you, just hit the back button, then Next again, and it may resolve itself. See this link for an explanation of the problem:

http://stackoverflow.com/questions/4382178/android-sdk-installation-doesnt-find-jdk


Android SDK Tools Setup

Java SE Development Kit

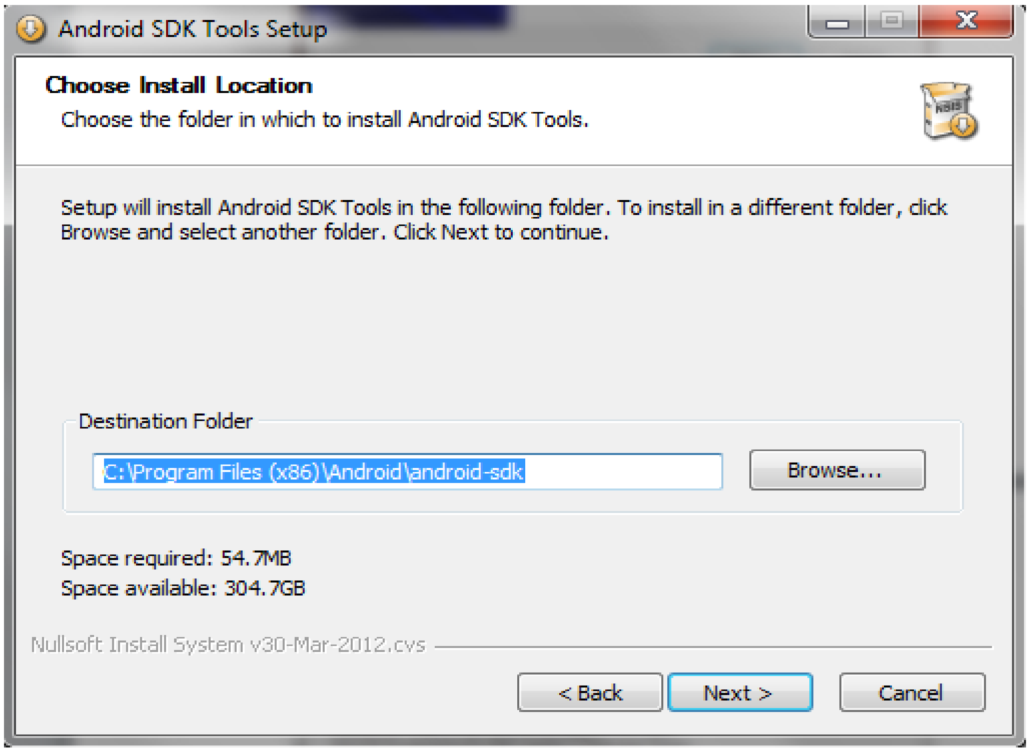
Detect whether Java SE Development Kit is installed.

Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.7 has been found.

Location: C:\Program Files (x86)\Java\jre7\bin\java.exe

Nullsoft Install System v30-Mar-2012.cvs

The install program has found the JDK, so we're good to go. Hit Next to proceed with the install process.


Android SDK Tools Setup

Choose Install Location

Choose the folder in which to install Android SDK Tools.

Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

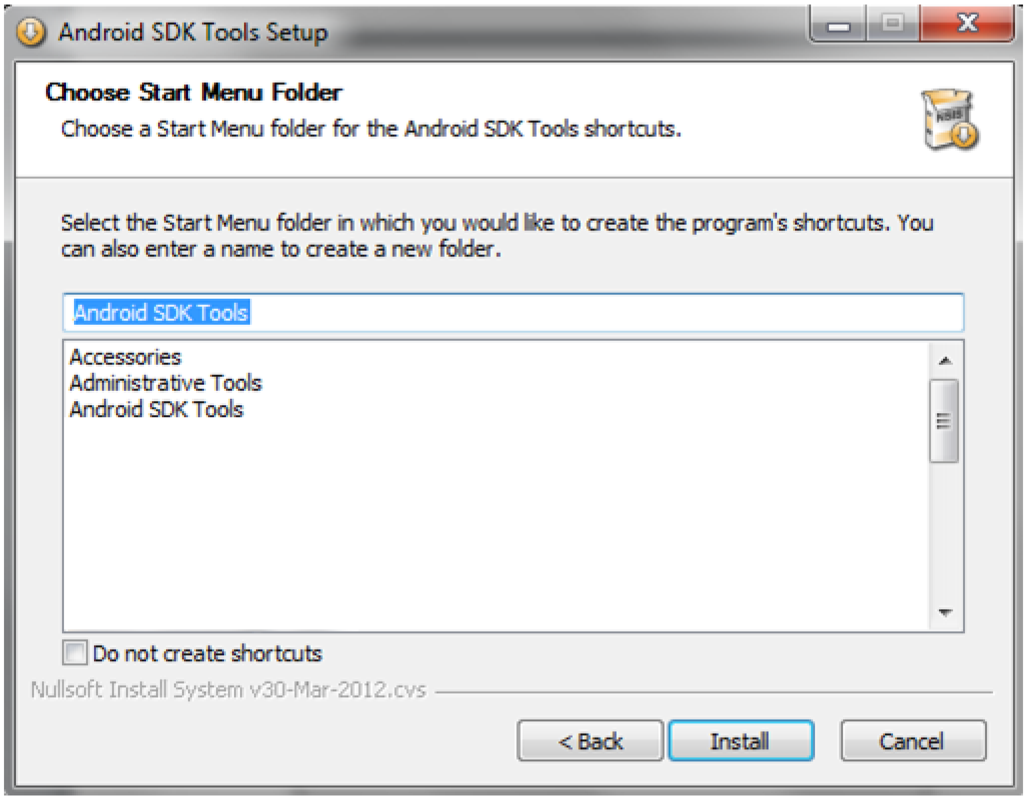
Destination Folder

C:\Program Files (x86)\Android\android-sdk

Space required: 54.7MB
Space available: 304.7GB

Nullsoft Install System v30-Mar-2012.cvs

In this screen you are asked to select the install location for the Android SDK. You can change it if you'd like, or retain the default. I choose to retain the defaults.


Android SDK Tools Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android SDK Tools shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

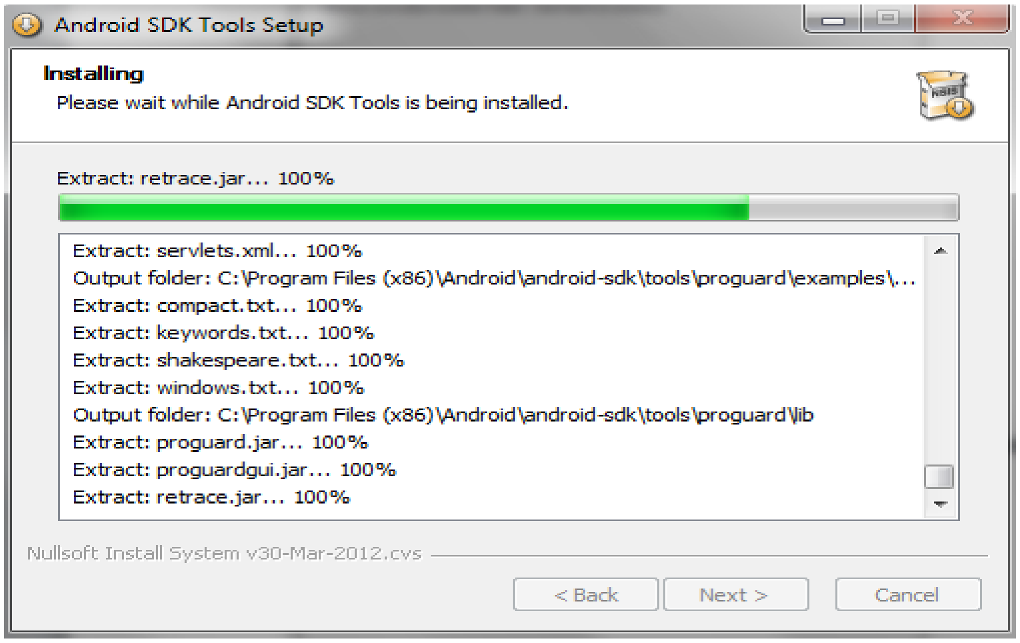
Android SDK Tools

Accessories
Administrative Tools
Android SDK Tools

[ ] Do not create shortcuts

Nullsoft Install System v30-Mar-2012.cvs
< Back
Install
Cancel

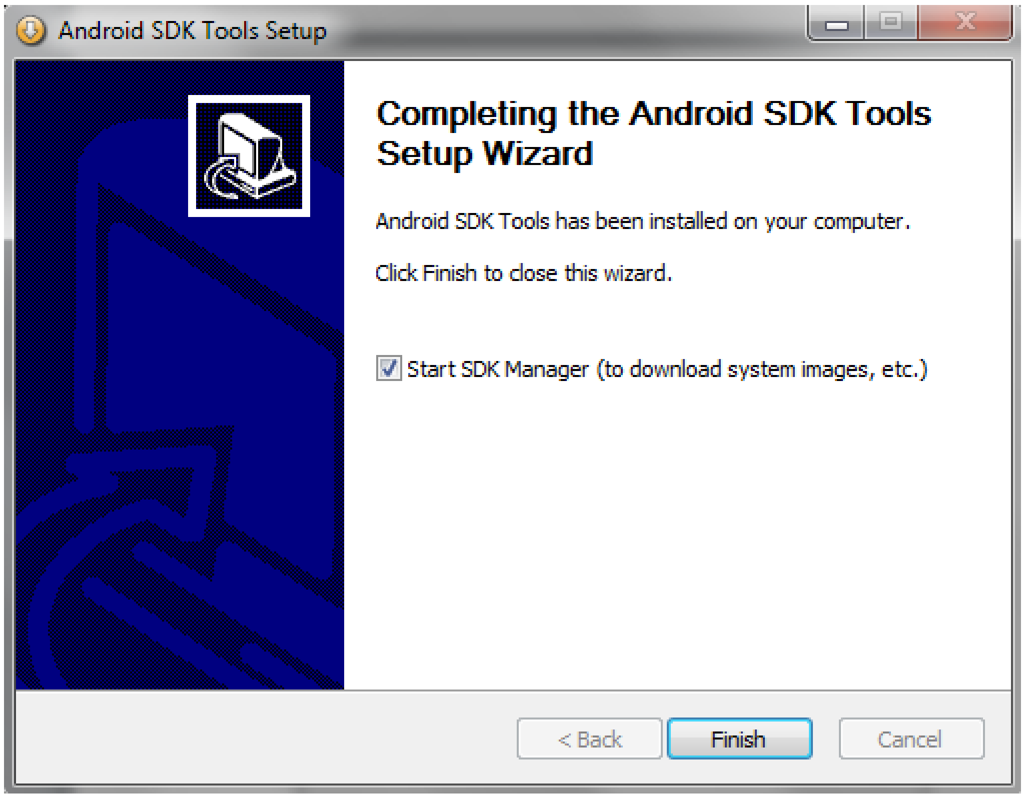
Now you're given the opportunity to create a Start Menu Folder for the Android SDK Tools. I generally accept the defaults here, and then hit **Install**.



```
Extract: retrace.jar... 100%
```

```
Extract: servlets.xml... 100%
Output folder: C:\Program Files (x86)\Android\android-sdk\tools\proguard\examples\...
Extract: compact.txt... 100%
Extract: keywords.txt... 100%
Extract: shakespeare.txt... 100%
Extract: windows.txt... 100%
Output folder: C:\Program Files (x86)\Android\android-sdk\tools\proguard\lib
Extract: proguard.jar... 100%
Extract: proguardgui.jar... 100%
Extract: retrace.jar... 100%
```

The install will go pretty quickly, and then you'll be presented with the following screen. You can, if you'd like, choose to **Start SDK Manager** and download the API's necessary for working with buzztouch. Since things are going well up to this point, let's keep rolling with it. Leave that box checked, and hit **Finish**.



Completing the Android SDK Tools Setup Wizard

Android SDK Tools has been installed on your computer.

Click Finish to close this wizard.

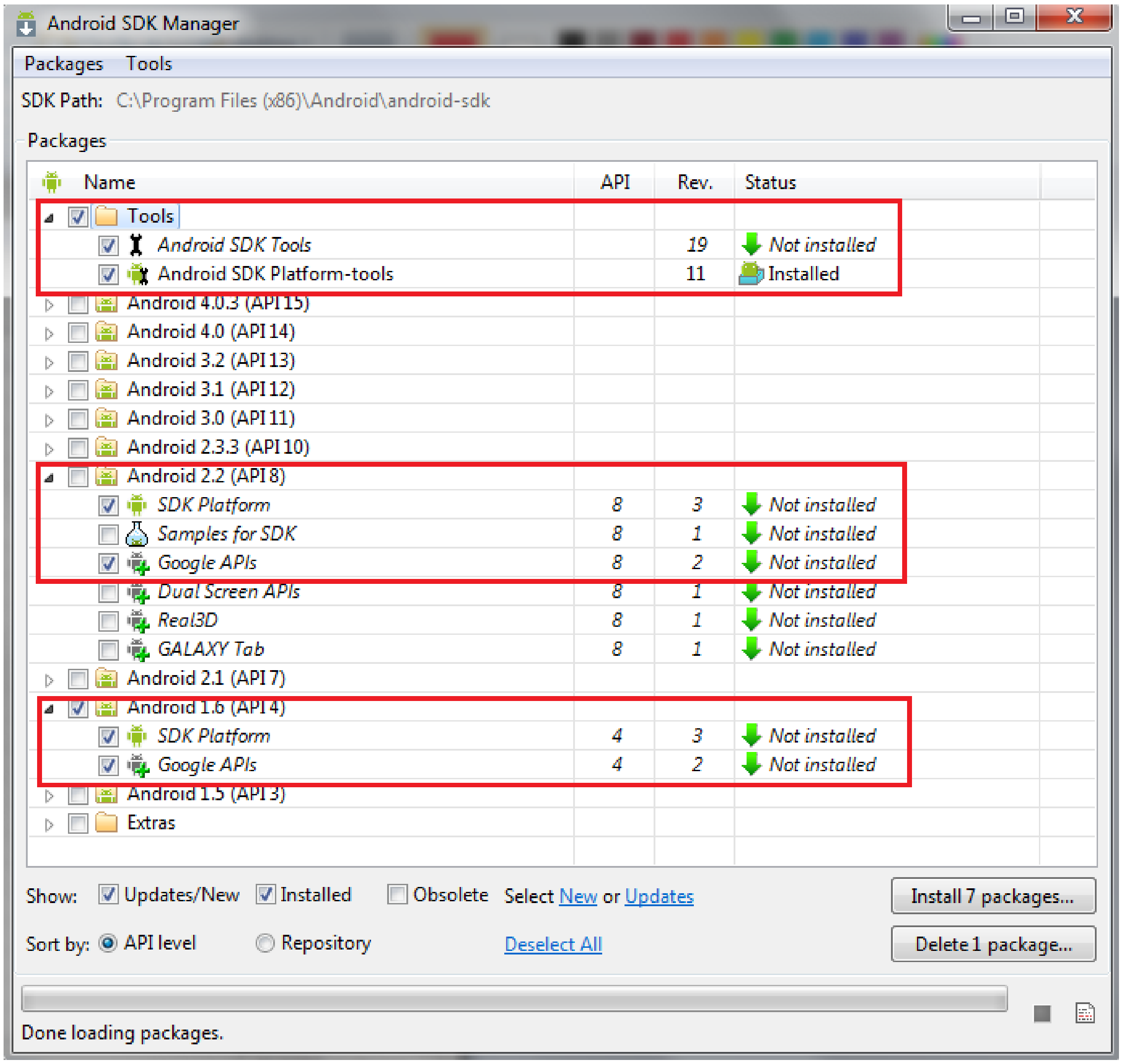
[✓] Start SDK Manager (to download system images, etc.)

4. **Configure the Android SDK for Development Tools and Google API**

This is one of the most important parts of setting up your Android development environment. It is during this procedure that you will be selecting, downloading and installing the Google APIs that are necessary for compiling and running your app. Android development is a bit confusing in that there are Android APIs, and Google APIs that can be used for development. The essential difference is that the Google APIs include the Google Maps API, which is necessary for Location Map screens. The Android API only includes core Android APIs, and does not support all the functionality that buzztouch provides. So, it's vitally important that for buzztouch v1.4, you select the Google APIs under Android 1.6 (API 4), and for buzztouch v1.5 and v2.0, you select the Google APIs under Android 2.2 (API 8). The **exact** items you need are highlighted below. Make sure your screen looks like the following one.

It should be noted that you can…and probably should…use Google APIs that are newer than the required ones. You just need to make sure your AndroidManifest.xml file reflects the requirements. But that's for another tutorial.

Remember: Google APIs and Android APIs **are not the same!** Don't make the mistake of thinking they are, or your app won't compile. This is one of the most common mistakes seen on the buzztouch forums.



Android SDK Manager

Packages Tools

SDK Path: C:\Program Files (x86)\Android\android-sdk

Packages

Name	API	Rev.	Status
[✓] Tools			
[✓] *Android SDK Tools*		*19*	*Not installed*
[✓] Android SDK Platform-tools		11	Installed
[ ] Android 4.0.3 (API 15)			
[ ] Android 4.0 (API 14)			
[ ] Android 3.2 (API 13)			
[ ] Android 3.1 (API 12)			
[ ] Android 3.0 (API 11)			
[ ] Android 2.3.3 (API 10)			
[ ] Android 2.2 (API 8)			
[✓] *SDK Platform*	*8*	*3*	*Not installed*
[ ] *Samples for SDK*	*8*	*1*	*Not installed*
[✓] *Google APIs*	*8*	*2*	*Not installed*
[ ] *Dual Screen APIs*	*8*	*1*	*Not installed*
[ ] *Real3D*	*8*	*1*	*Not installed*
[ ] *GALAXY Tab*	*8*	*1*	*Not installed*
[ ] Android 2.1 (API 7)			
[✓] Android 1.6 (API 4)			
[✓] *SDK Platform*	*4*	*3*	*Not installed*
[✓] *Google APIs*	*4*	*2*	*Not installed*
[ ] Android 1.5 (API 3)			
[ ] Extras			

Show: [✓] Updates/New [✓] Installed [ ] Obsolete Select New or Updates

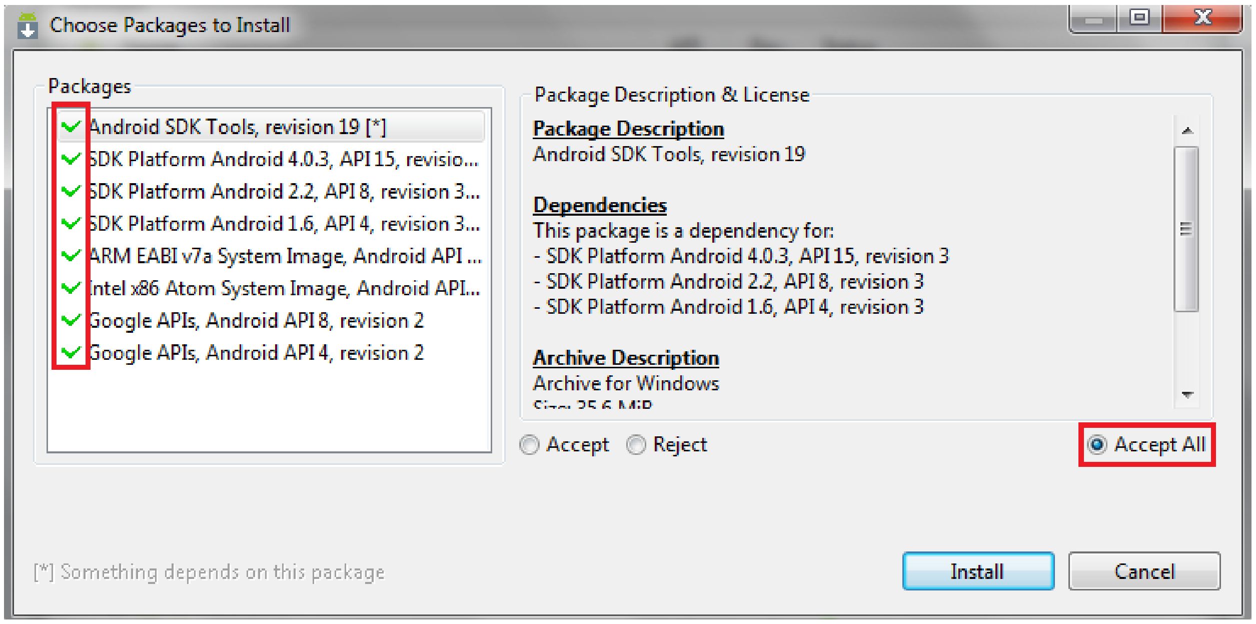
Install 7 packages...

Sort by: [● API level] ○ Repository Deselect All

Delete 1 package...

Done loading packages.

Once you have the correct items selected as noted above, hit **Install x packaged** (where x is however many you have), and then you'll get the following screen. Hit the **Accept All** radio button so that all the packages listed have a green checkmark next to them. You're essentialy accepting licenses for these packages here…nothing worth reading!


Choose Packages to Install

Packages

- ✓ Android SDK Tools, revision 19 [*]
- ✓ SDK Platform Android 4.0.3, API 15, revisio...
- ✓ SDK Platform Android 2.2, API 8, revision 3...
- ✓ SDK Platform Android 1.6, API 4, revision 3...
- ✓ ARM EABI v7a System Image, Android API ...
- ✓ Intel x86 Atom System Image, Android API...
- ✓ Google APIs, Android API 8, revision 2
- ✓ Google APIs, Android API 4, revision 2

Package Description & License

Package Description
Android SDK Tools, revision 19

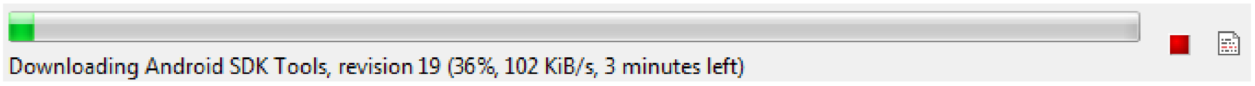
Dependencies
This package is a dependency for:
- SDK Platform Android 4.0.3, API 15, revision 3
- SDK Platform Android 2.2, API 8, revision 3
- SDK Platform Android 1.6, API 4, revision 3

Archive Description
Archive for Windows

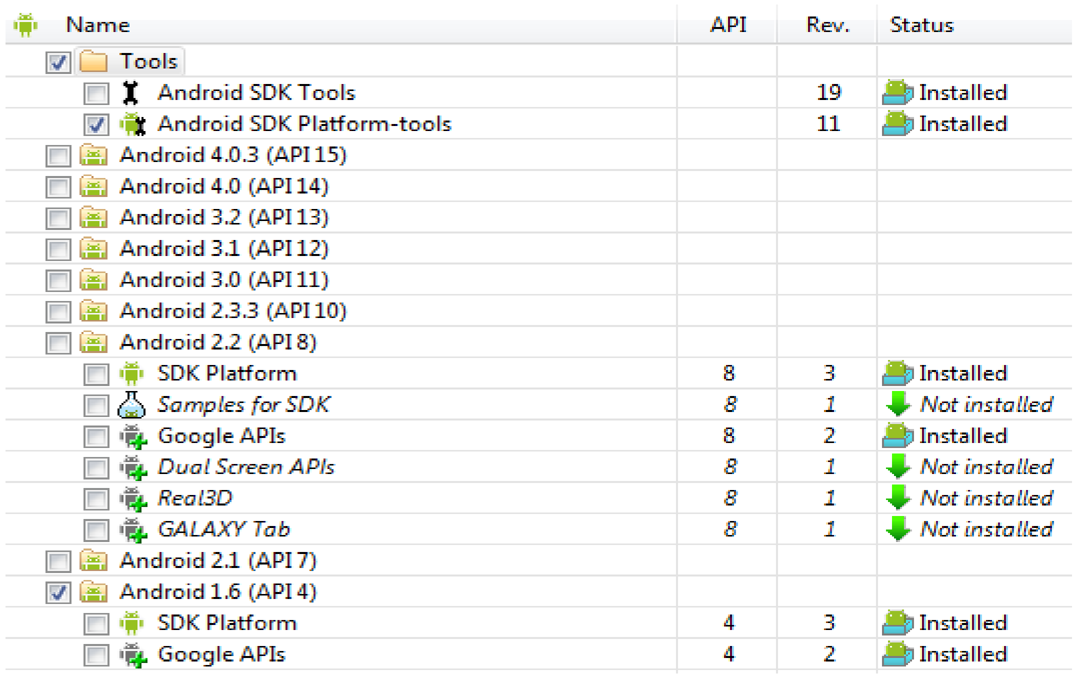
Accept Reject [✓ Accept All]

[*] Something depends on this package

Now that you've hit install, you'll see a progress indicator. Even with a fast Internet connection, this is going to take quite some time. So, take a few minutes…hop online and order me a case of Fruit Punch SoBe…and before you know it, the green progress indicator you see below will be all the way full!


Downloading Android SDK Tools, revision 19 (36%, 102 KiB/s, 3 minutes left)

Pretty soon you'll see a "Downloading packages complee" notification, and you're done! Verify that all the packages you want show as installed.

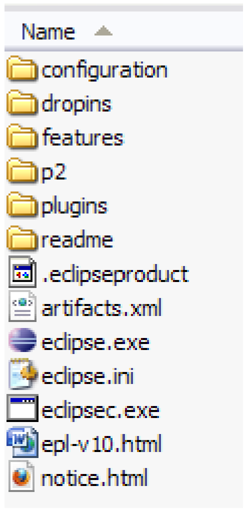


	Name	API	Rev.	Status
✓	Tools			
	Android SDK Tools		19	Installed
✓	Android SDK Platform-tools		11	Installed
	Android 4.0.3 (API 15)			
	Android 4.0 (API 14)			
	Android 3.2 (API 13)			
	Android 3.1 (API 12)			
	Android 3.0 (API 11)			
	Android 2.3.3 (API 10)			
	Android 2.2 (API 8)			
	SDK Platform	8	3	Installed
	Samples for SDK	*8*	*1*	*Not installed*
	Google APIs	8	2	Installed
	Dual Screen APIs	*8*	*1*	*Not installed*
	Real3D	*8*	*1*	*Not installed*
	GALAXY Tab	*8*	*1*	*Not installed*
	Android 2.1 (API 7)			
✓	Android 1.6 (API 4)			
	SDK Platform	4	3	Installed
	Google APIs	4	2	Installed

You've now completed the installation and configuration of the Android SDK. Congratulations! Time to move on to the next step!

5. Unzip and move the Eclipse IDE to the desired location

This is the easiest thing you'll do today. Simply unzip the package you downloaded into the folder of your choice. I recommend **c:\eclipse.** Your folder structure should now basically look like what you see below. The **eclipse.exe** file is what you launch to start the program. Feel free to create a shortcut on your desktop, or pin it to your Start Menu (which I did). You'll want to run this program as Administrator.



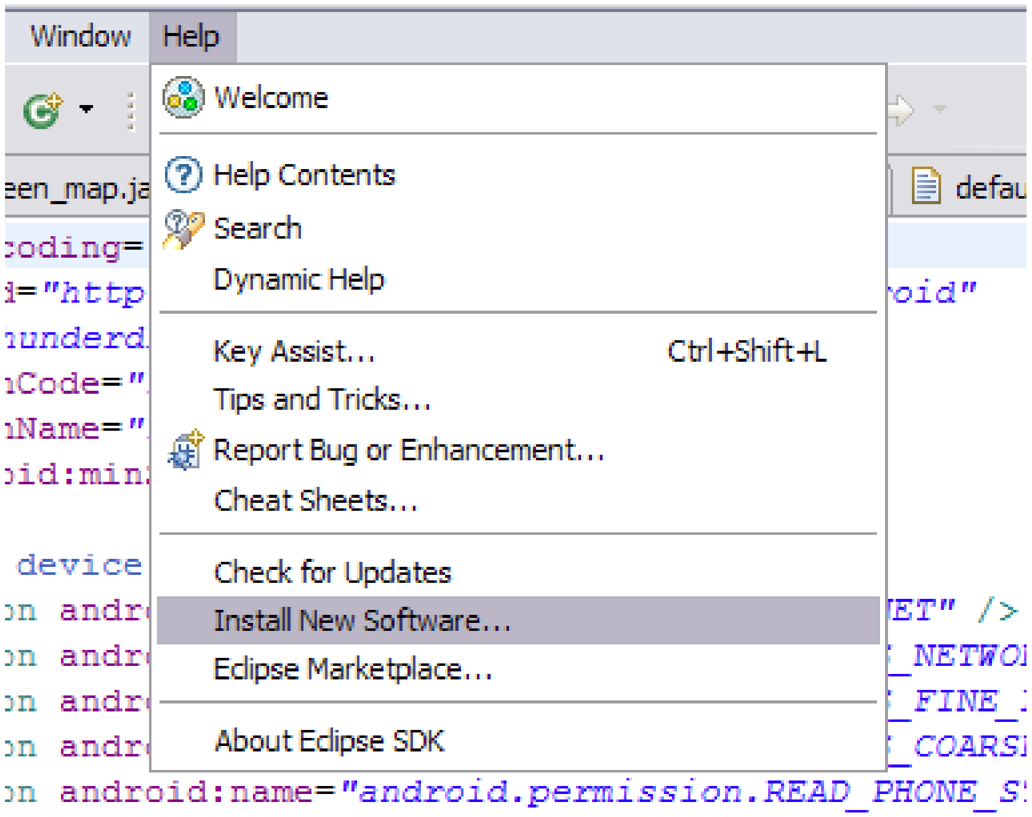
```
Name
configuration
dropins
features
p2
plugins
readme
.eclipseproduct
artifacts.xml
eclipse.exe
eclipse.ini
eclipsec.exe
epl-v10.html
notice.html
```

See, I told you that was easy! If you're stuck because you're not sure how to unzip something, I recommend heading to http://winrar.soft32.com/free-download and downloading WinRAR. It allows for right-clicking on a zip file and deciding where to unzip it to. Cool stuff!

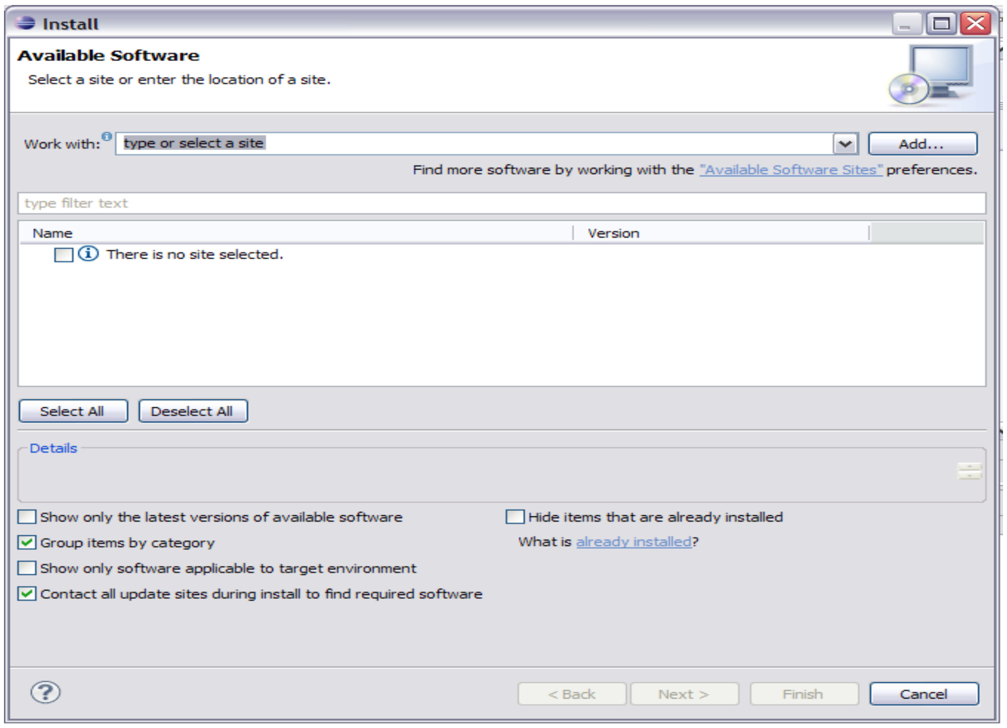
6. Install the ADT Plugin for Eclipse

We now need to install the Android Development Tools for Eclipse. This is a pretty simple process, and doesn't take too long. It'll give you the extra few things you need for developing Android applications (remember that Eclipse is an IDE capable of developing many, many things). But without these tools, you simply cannot tie together the Android SDK and Eclipse for use in developing apps.

Start Eclipse by double-clicking on the Eclipse icon you created earlier. After Eclipse has launched go to **Help…Install New Software**



This will bring up the following install dialogue box. You'll see that there is no site selected, and thus no option on which software to install.


Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☐ There is no site selected.	

Details

☐ Show only the latest versions of available software

☑ Group items by category

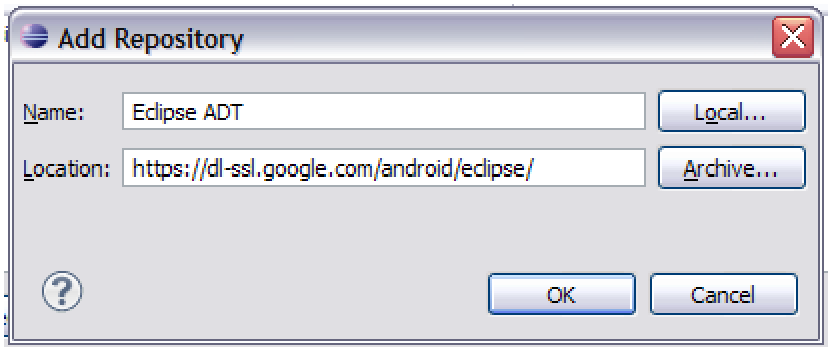
☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

☐ Hide items that are already installed

What is already installed?

Click the **Add…** button and the you’ll get this:

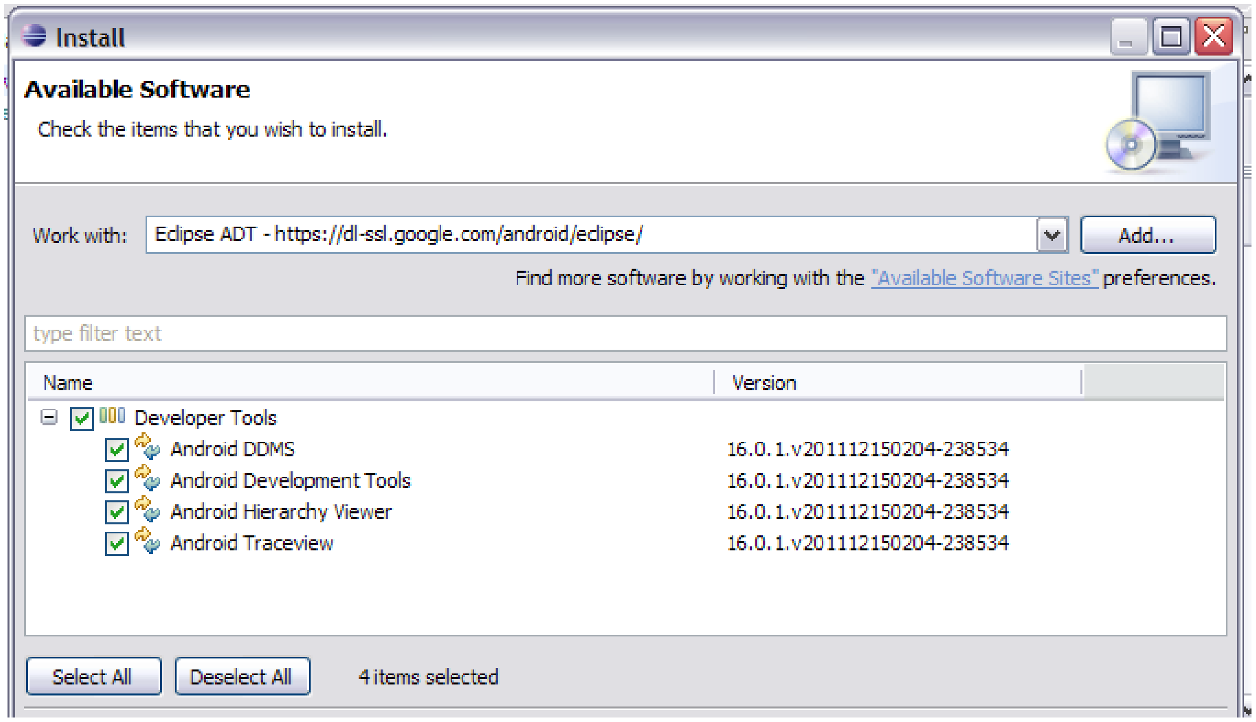

Add Repository

Name: Eclipse ADT

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

Specify the same information you see above. For clarity, the link you want is:

https://dl-ssl.google.com/android/eclipse. You can give it whatever name you want. Hit **OK**. Once you’ve done that, you’ll get the following screen:


Install

Available Software

Check the items that you wish to install.

Work with: Eclipse ADT - https://dl-ssl.google.com/android/eclipse/

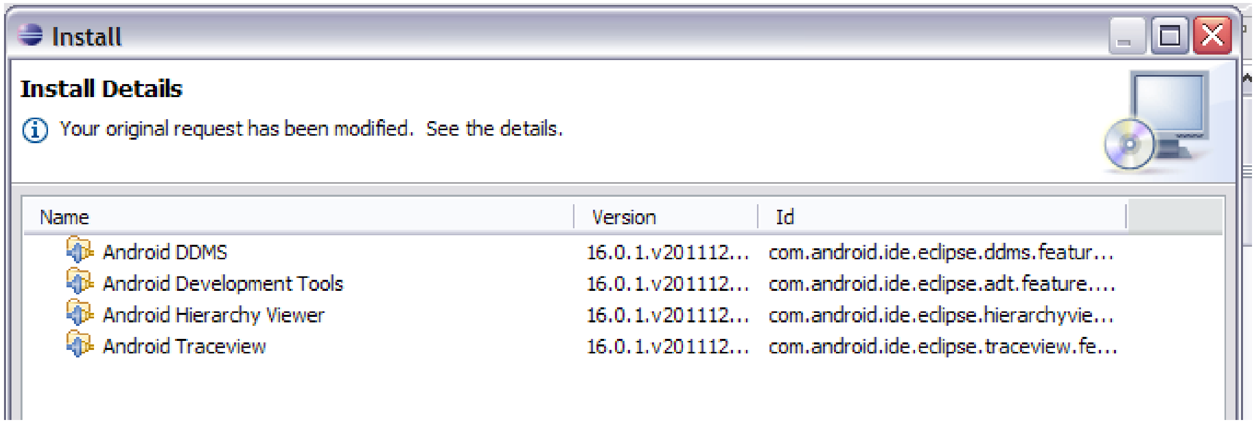
Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Developer Tools	
☑ Android DDMS	16.0.1.v201112150204-238534
☑ Android Development Tools	16.0.1.v201112150204-238534
☑ Android Hierarchy Viewer	16.0.1.v201112150204-238534
☑ Android Traceview	16.0.1.v201112150204-238534

4 items selected

Well, that's half the screen, but you get the picture! These are your options for install, and it includes the Android Development Tools (ADT). I went ahead and selected everything figuring I may need some of the other stuff in the future. Hit **Next**.

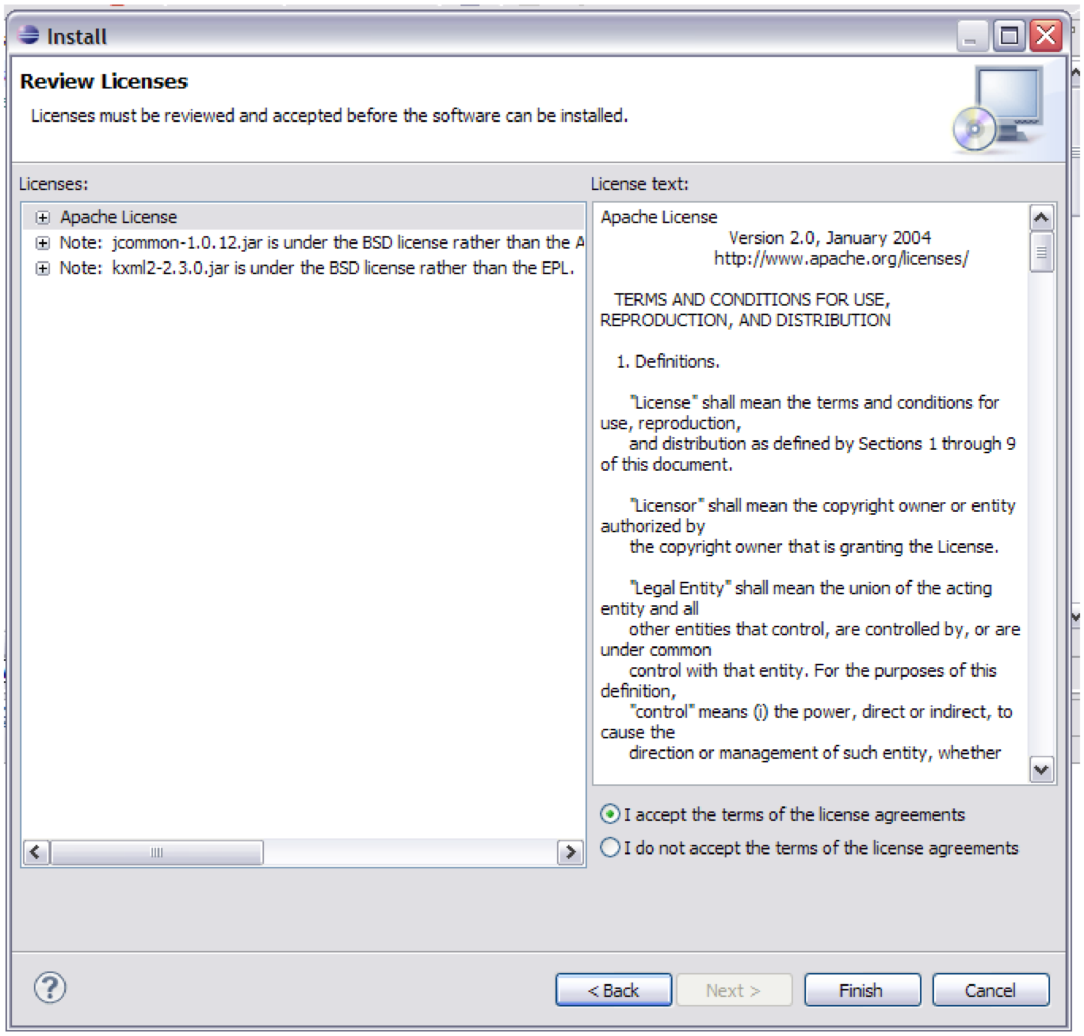

Install

Install Details

Your original request has been modified. See the details.

Name	Version	Id
Android DDMS	16.0.1.v201112...	com.android.ide.eclipse.ddms.featur...
Android Development Tools	16.0.1.v201112...	com.android.ide.eclipse.adt.feature....
Android Hierarchy Viewer	16.0.1.v201112...	com.android.ide.eclipse.hierarchyvie...
Android Traceview	16.0.1.v201112...	com.android.ide.eclipse.traceview.fe...

This screen verifies the details of what you are going to install. Hit **Next**.


Install

Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

- Apache License
- Note: jcommon-1.0.12.jar is under the BSD license rather than the A
- Note: kxml2-2.3.0.jar is under the BSD license rather than the EPL.

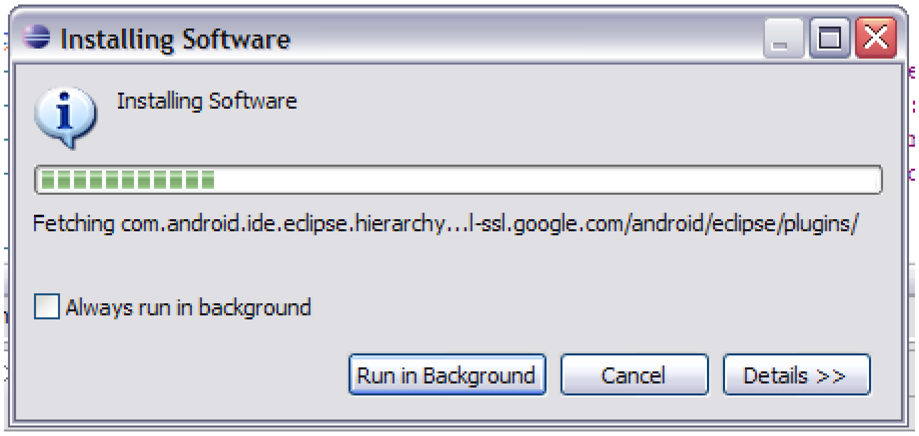
License text:

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE,
REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for
use, reproduction,
      and distribution as defined by Sections 1 through 9
of this document.

      "Licensor" shall mean the copyright owner or entity
authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting
entity and all
      other entities that control, are controlled by, or are
under common
      control with that entity. For the purposes of this
definition,
      "control" means (i) the power, direct or indirect, to
cause the
      direction or management of such entity, whether
```

[✓ I accept the terms of the license agreements]

I do not accept the terms of the license agreements

If you feel the need, review the licenses for the stuff you are going to install here. Most important point on this screen is to select **I accept the terms of the license agreements**. If you don't, you won't be able to install. Hit **Next**.

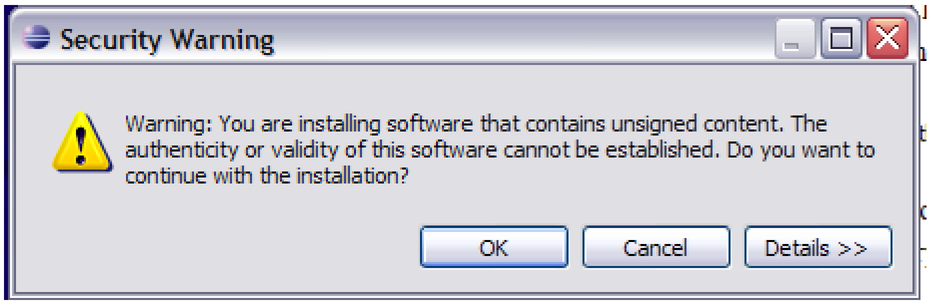

Installing Software

Installing Software

Fetching com.android.ide.eclipse.hierarchy...l-ssl.google.com/android/eclipse/plugins/

Always run in background

Congratulations! ADT is now installing! Sit back and relax…at least until you see this screen:


Security Warning

Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

Nothing to be worried about here. Just hit **OK** and move on. It's all good. After install is done, the program will ask if you would like to reset Eclipse so the changes can take effect. Say **Reset Eclipse**, wait for it to restart, and you can move on to the next step.

At this point, you are basically done, unless you want to create an Android Virtual Device (AVD), or emulator, as it's often called. If you're ready for that, head on over to **Chapter 6: Configuring Your Android Virtual Device (AVD).**

Congratulations on making it through one of the more difficult parts of developing for Android! Hopefully this chapter has made it as painless a process as possible.

Chapter 6

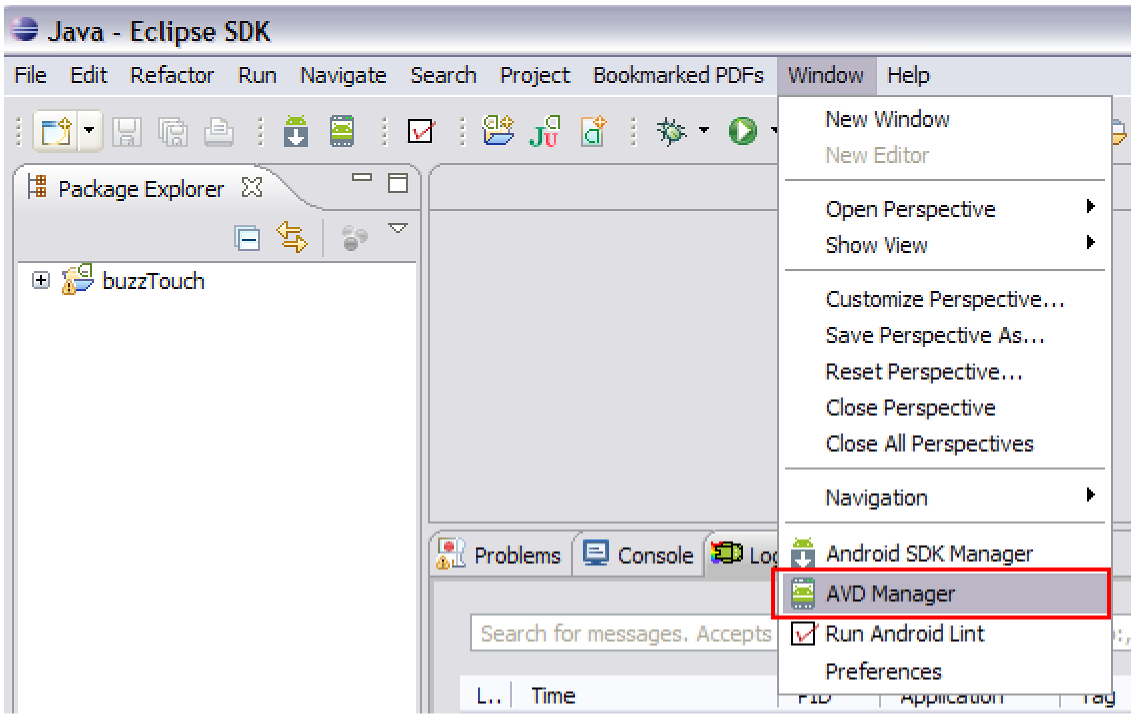
Configuring Your Android Virtual Device (AVD)

Now that you have the Android development environment set up and configured, you're ready to start building and testing your programs. But, while it's preferable to test your app on an actual device, it's possible that you don't have one. In comes the Android Virtual Device to the rescue!

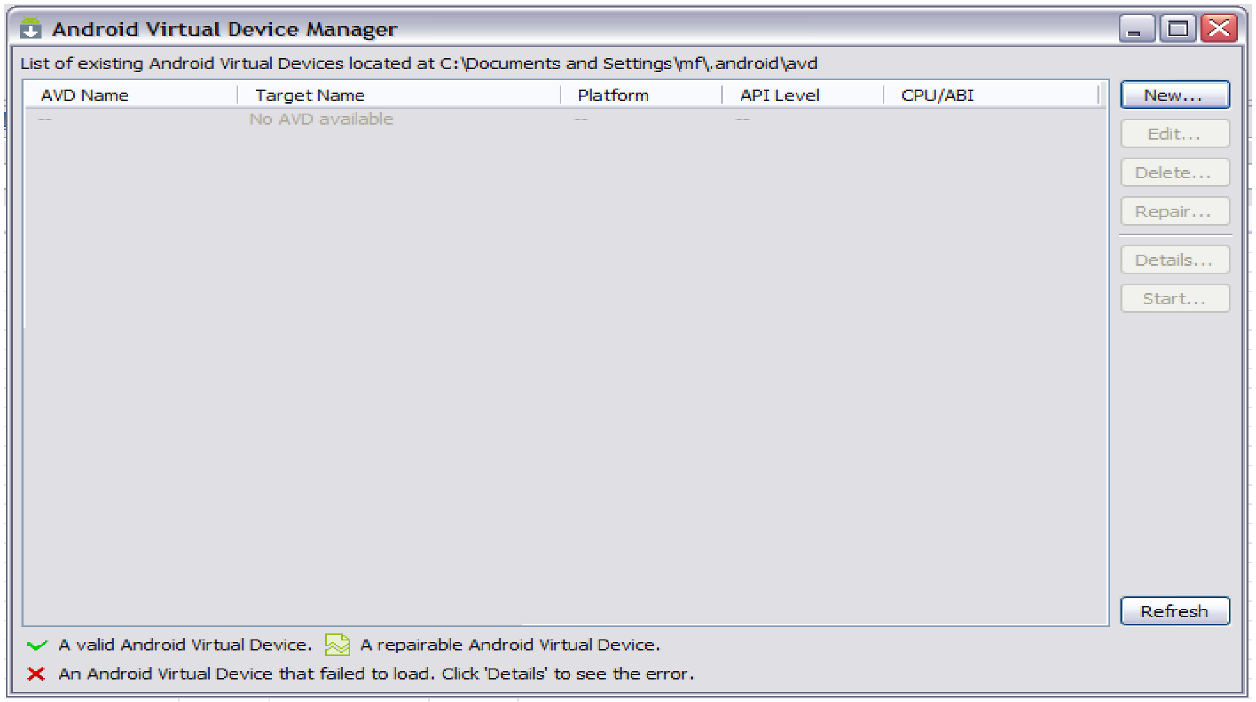
Keep in mind, however, that the AVD is an emulator…it emulates what an actual device would do. And while it is super useful in helping you test and debug your app, it cannot do some things that a real device can do like use GPS to get real coordinates. So, there will be limitations to what you can accomplish with the AVD, but it you keep that in mind, and also try to find a real device to test on, this can still be a super valuable weapon in your development arsenal.

Enough blather…let's get on to creating an AVD!

To begin, go to **Windows…AVD Manager.**



This will bring up the AVD Manager screen, which, since you currently have no AVDs configured, will look like this:



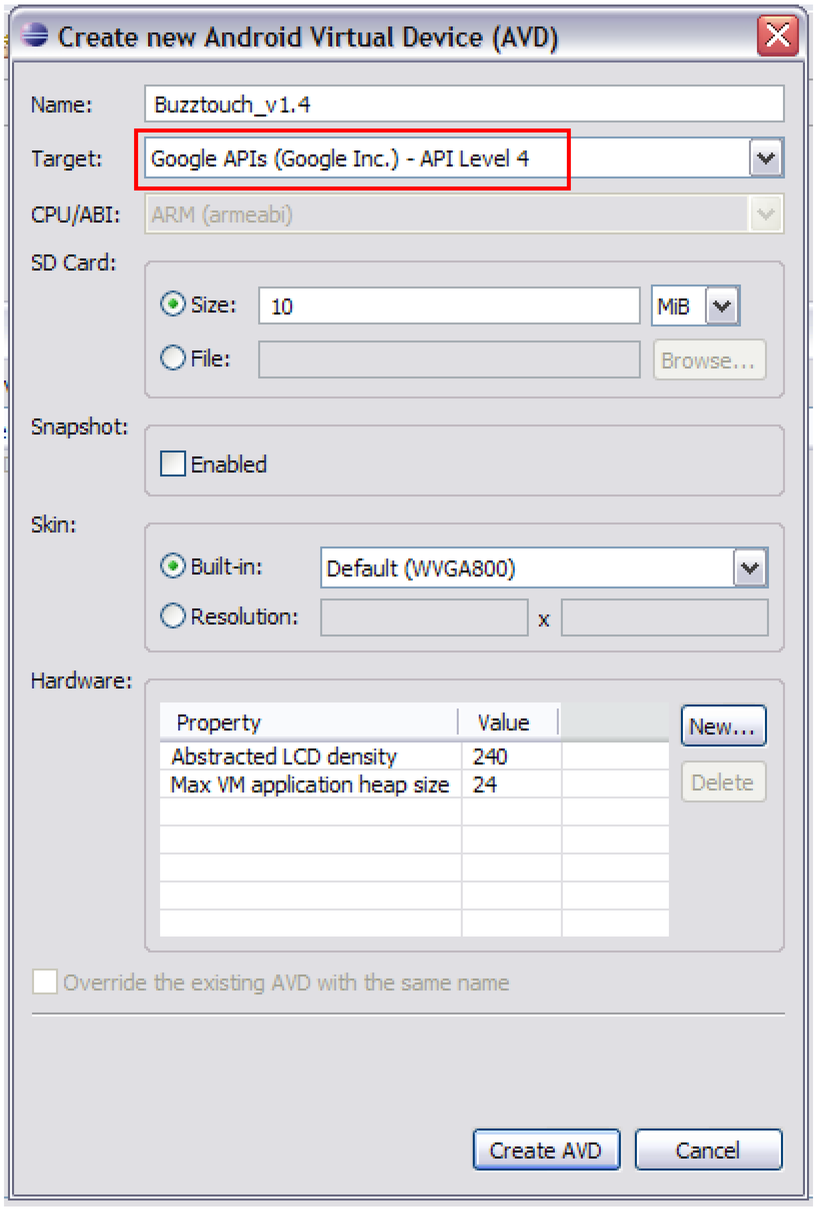
List of existing Android Virtual Devices located at C:\Documents and Settings\mf\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

There are two different AVDs that you could create…one for buzztouch v1.4, and one for buzztouch v1.5/v2 Since they both use different Google APIs, You'll have to create one for each if you develop for both. If you are only doing one or the other, then you only need to create an AVD for that particular version of buzztouch.

In order to create an AVD for buzztouch v1.4, click **New…** and fill in the information like you see below:


Create new Android Virtual Device (AVD)

Name: Buzztouch_v1.4

Target: Google APIs (Google Inc.) - API Level 4

CPU/ABI: ARM (armeabi)

SD Card: Size: 10 MiB

Snapshot: Enabled [ ]

Skin: Built-in: Default (WVGA800)

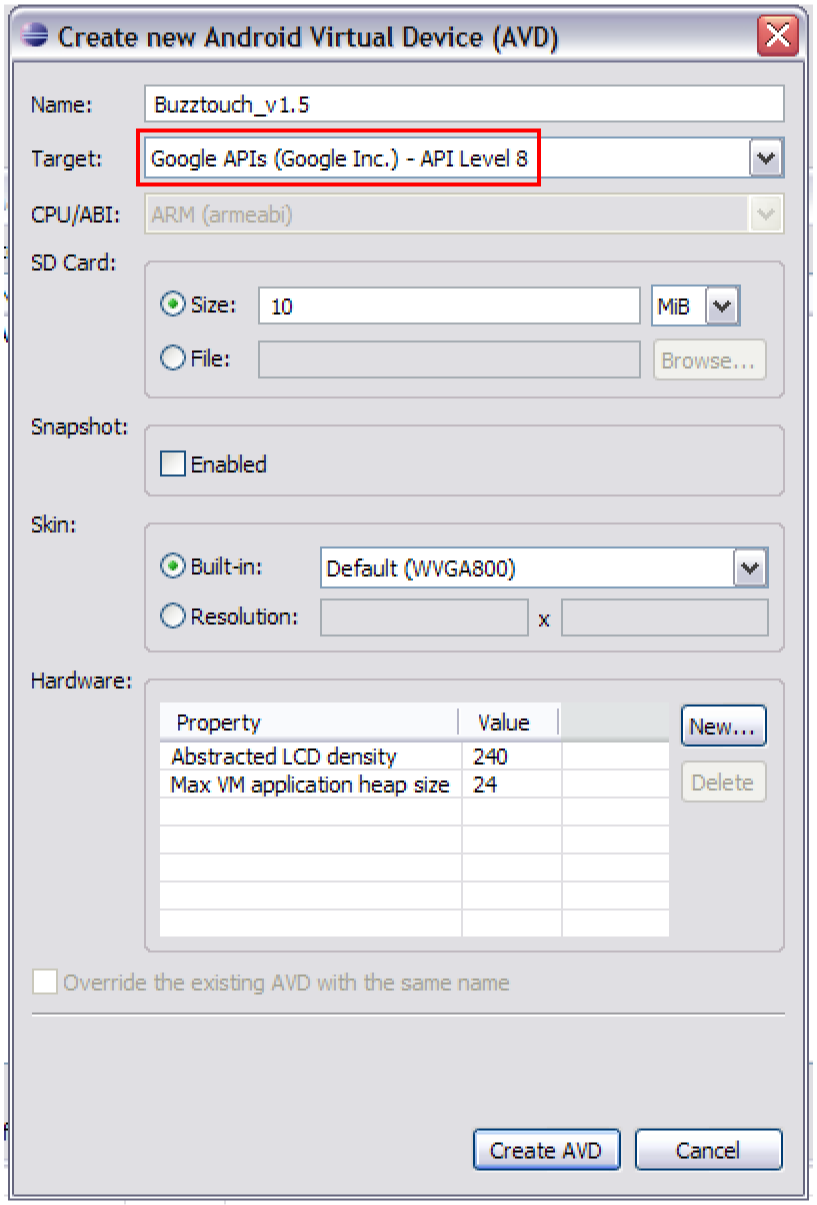
Hardware:

Property	Value
Abstracted LCD density	240
Max VM application heap size	24

File:
Browse...
Resolution:
New...
Delete
Override the existing AVD with the same name
Create AVD
Cancel

You can adjust the size of the SD Card to your liking, but I choose to keep it near the minimum 9MB, simply because I'm not using the SD Card feature, and because it loads faster with a smaller SD Card configured. I also leave the default Built-in skin of WVGA800, but you can change the resolution to a device size of your liking.

In order to create an AVD for buzztouch v1.5/BT Server 2.0, click **New…** and fill in the information like you see below:


Create new Android Virtual Device (AVD)

Name: Buzztouch_v1.5

Target: Google APIs (Google Inc.) - API Level 8

CPU/ABI: ARM (armeabi)

SD Card: [✓ Size:] 10 MiB

File:

Snapshot: Enabled

Skin: [✓ Built-in:] Default (WVGA800)

Resolution: x

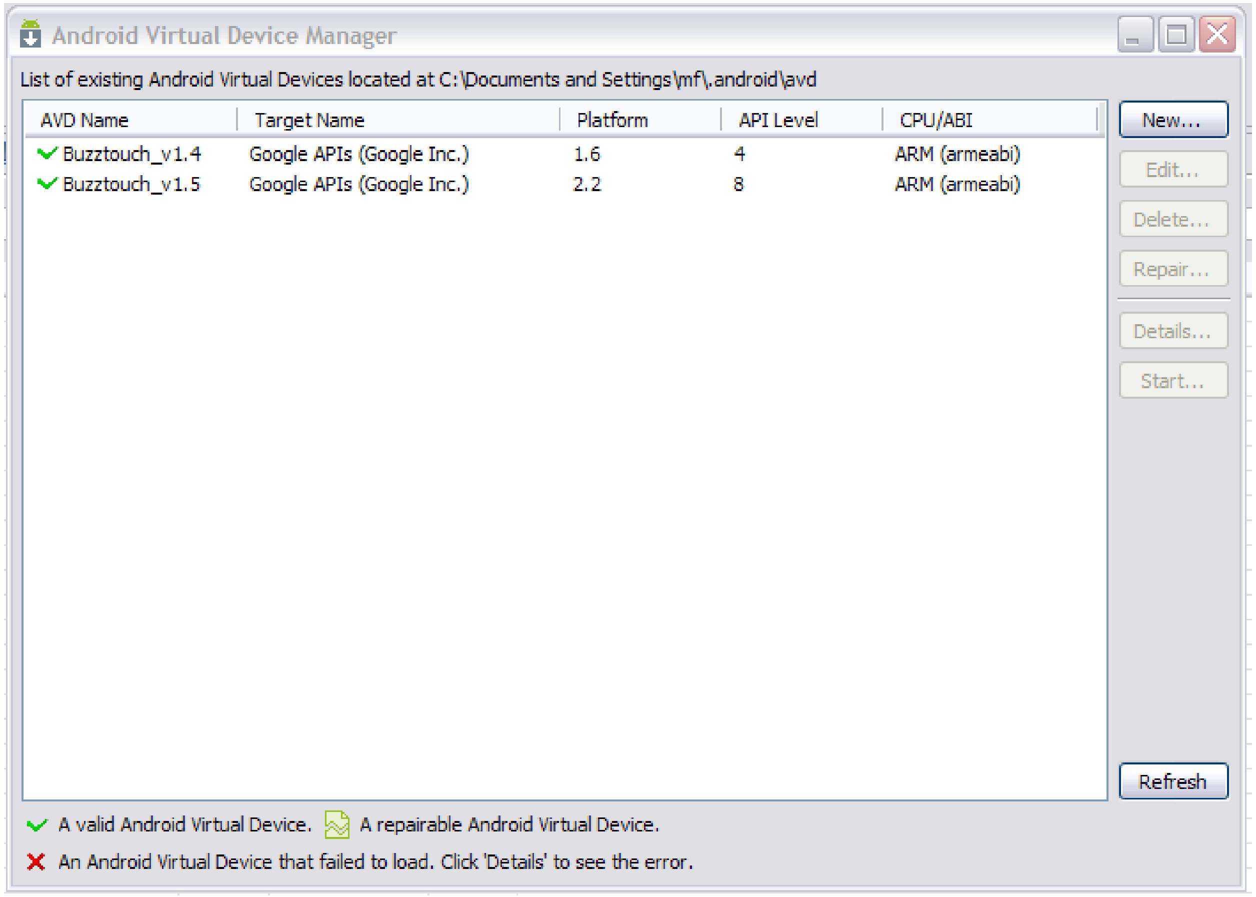
Hardware:

Property	Value
Abstracted LCD density	240
Max VM application heap size	24

Override the existing AVD with the same name

You can adjust the size of the SD Card to your liking, but I choose to keep it near the minimum 9MB, simply because I'm not using the SD Card feature, and because it loads faster with a smaller SD Card configured. I also leave the default Built-in skin of WVGA800, but you can change the resolution to a device size of your liking.

Now that you are done creating your virtual devices, your Android Virtual Device manager should look something like what you see below. The important thing is that the correct Google APIs are selected for the version of buzztouch that you are using, and that there is a green checkmark next to each virtual device, indicating that it is good.


Android Virtual Device Manager

List of existing Android Virtual Devices located at C:\Documents and Settings\mf\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Buzztouch_v1.4	Google APIs (Google Inc.)	1.6	4	ARM (armeabi)
✓ Buzztouch_v1.5	Google APIs (Google Inc.)	2.2	8	ARM (armeabi)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

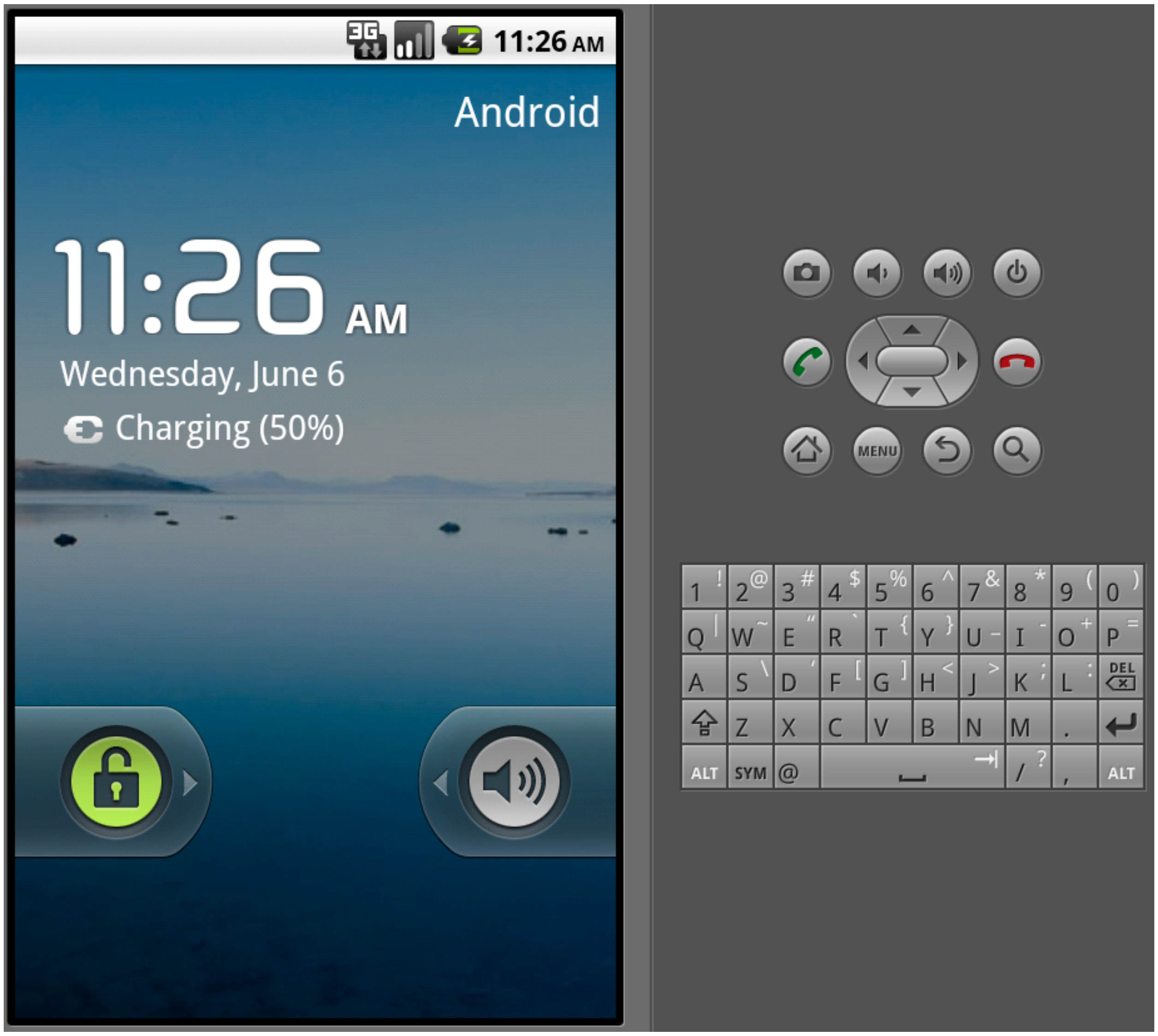
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

At this point you are done defining your AVD. You can launch an AVD by selecting it from the panel above, and hitting the **Start** button. Or, you can open up a command prompt (Windows) or terminal window (Mac), navigate to the **tools** folder of your Android SDK installation, where you will see various emulator executables. Then you can type in the following command to launch your emulator (launching the v1.5 emulator in this example):

```
emulator @buzztouch_v1.5
```

Launching the emulator is super, super, super slow, and sometimes times out if you are trying to do it from within Eclipse and install your app at the same time. There is no known fix to this yet. Sometimes you'll just have to exit the emulator and start again!

Depending on the version of the emulator that you started, you should get a screen that looks like the following. To unlock, using your mouse, just pull the lock graphic to the right!



At this point, just start playing around with it to see what it can do. The buttons on the right side of the emulator represent keys on a device (usually a phone). The Home (house) key will take you to the main screen. The Menu key will present you with any menu options for your current screen (and is a way to refresh your app). The Arrow button will take you back a screen.

Have fun playing around! There's lots you can do here, and at the very least, you should test each and every one of your Android apps on the emulator!

Chapter 7

Compiling and Running Your App

The time has come for you to actually run and potentially use that Android app you have spent so many hours creating at buzztouch! Your environment is ready…you have an Emulator all set up and ready to go…and you've downloaded your source code from buzztouch, so you're ready to compile your app. Cool!

Wait…what? Source code? Compile? What the heck am I talking about, you ask?!

This is a slight digression, but something I think worth spending a few words on. The buzztouch website allows you to define and connect the screens and content for an Android application. This is done through screen creation, media uploads, a method to link it all together as well as the ability to tweak all kinds of app parameters (background color, email functionality, GPS, etc). Some people might think that after the app has been "designed" on buzztouch, it's ready to download and send to the store.

Not quite.

What buzztouch gives you after you have designed your app is a package of files – source code – that can be used to build the final app. That source code contains all the necessary information to create an app with the screens you described, as well as interact with the user in the way you intend. But, by itself, the source code can't do anything. That's where Eclipse and the Android SDK come in. Using your Eclipse environment, you will import your project from buzztouch, compile your code (translate the code into instructions the device can understand), sign your app (so people can be confident it really came from you), and create the apk file (which is the final "binary" that you load on your device, or submit to Google Play). Seems a bit complicated, but it's not. That's all you really need to understand. So, let's get started. We'll start by showing you how to download your source code, and then look at two methods for getting your project into Eclipse and exiting with an apk.

Downloading Source Code

As we discussed above, you'll need your source code package in order to compile your all. Getting that is pretty simple! Here are the steps:

1. Log into buzztouch.com
2. Click on your account name
3. Click on Applications
4. In the list of applications you are presented with (assuming you have created one!), click on the application you want source code for

5. Click “Download Project Source Code”
6. Click “Prepare Package For Download” (Android option is on the right side of the screen)
7. Watch it prepare your package and be amazed at the process!
8. Click “Download zip archive (x.x MB)”
9. Save the package to a **location you will remember!**
10. Go to the zipped package and unzip it using a program like WinRAR for Windows, or Archive Utility for Mac.

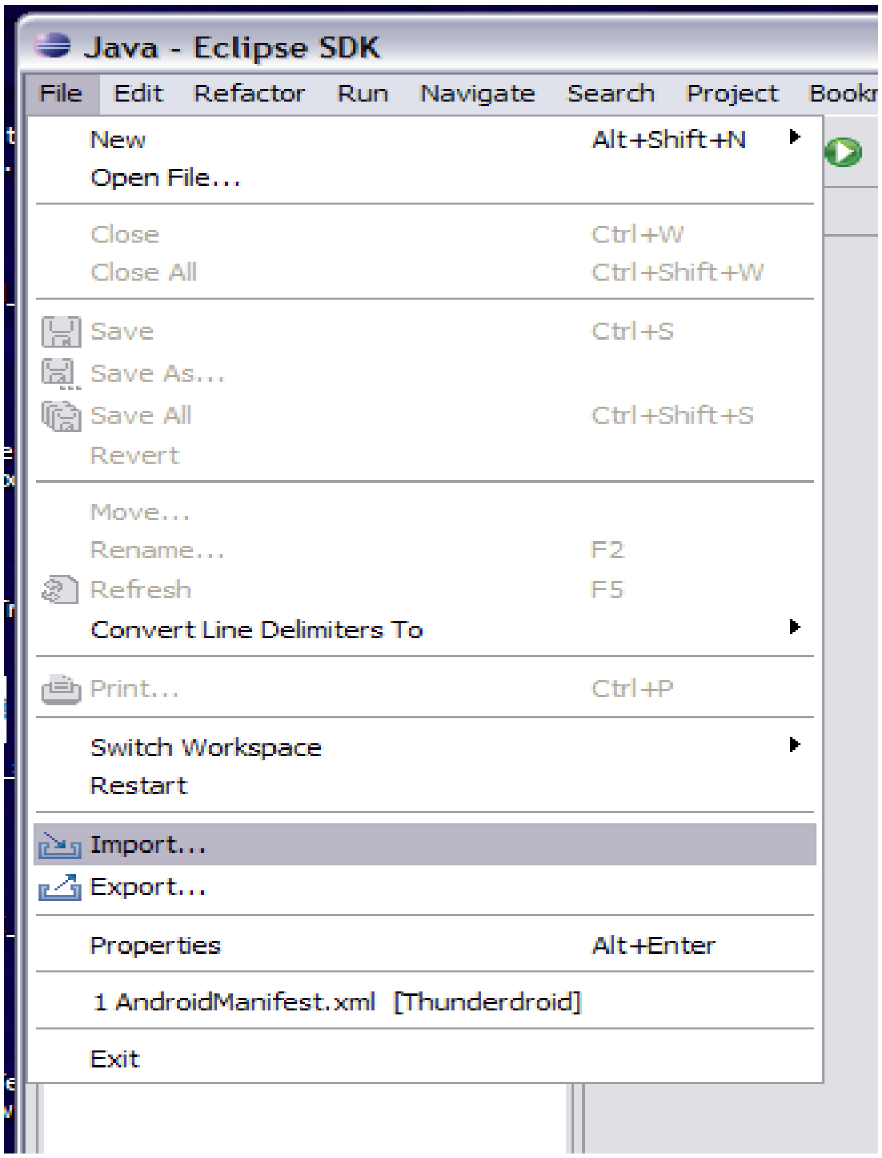
You now have an unzipped copy of your source code. If you are interested, take some time and look around the contents…you can’t hurt anything by just looking! You’ll see a number of folders, many of them with various .java files in them. This is the code that actually does the things that you want your app to do (but you know that anyway, since you’ve completed buzztouch U by this point!). You’ll also see, in the **assets** folder your BT_Config.txt file. This is the JSON file that literally defines almost every aspect of your app…something else you already know from doing buzztouch U (getting the hint here!?). Lots of stuff to explore in these packages, so take some time, check things out, and try to get a better understanding of what’s going on under the hood.

OK…time to bring your project into Eclipse and get some work done!

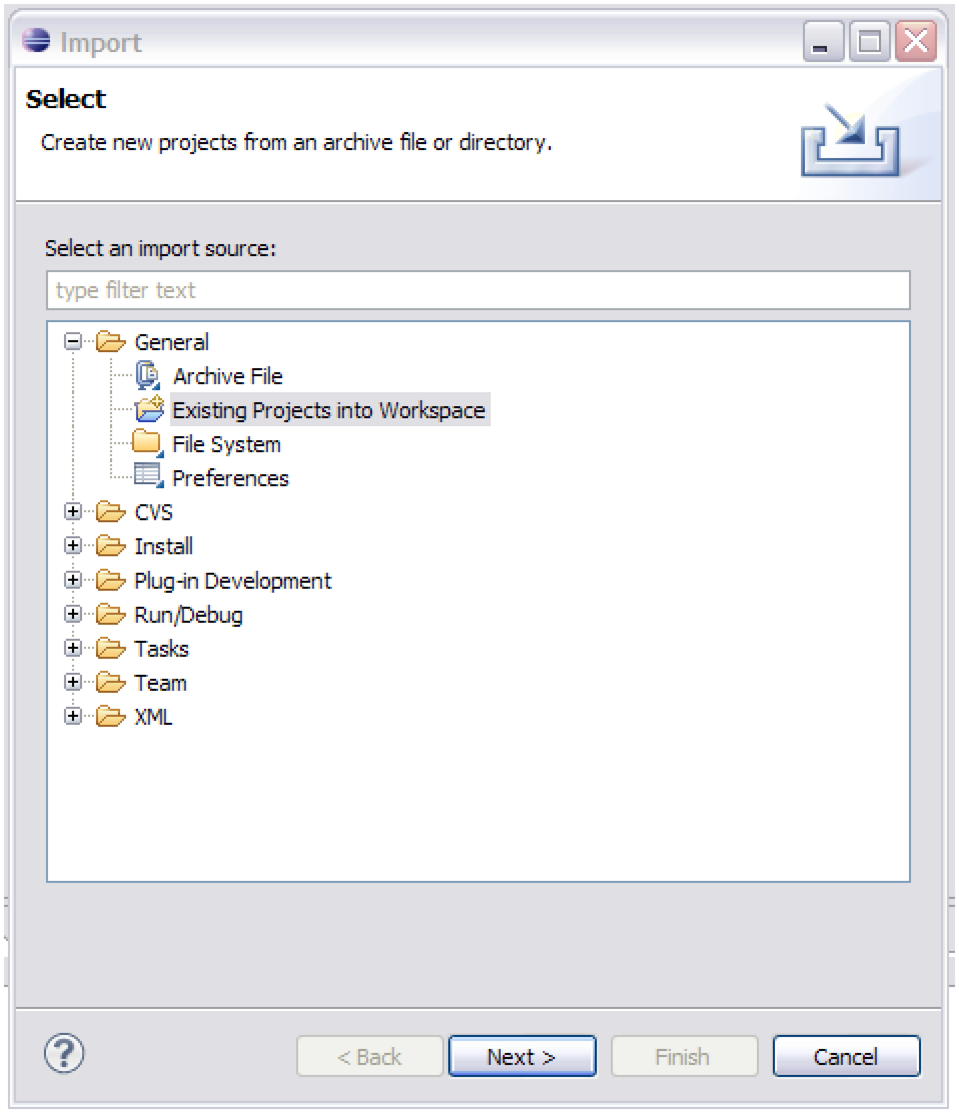
Method One: Import Your Project

When you download your Android code package from buzztouch, you’re essentially getting a complete project. All the files you need are contained in the zipped folder you get. So, importing your project into Eclipse is as easy as specifying the folder location, proving a bit of info, and hitting OK! Let’s get started!

1. **Start Eclipse**. Start Eclipse by either double-clicking on the icon you set up for it somewhere, or double-clicking on the eclipse.exe file that is found in your installation path. For me, it’s at c:\eclipse\eclipse.exe (on a Windows machine).
2. **Import Your Project.**
    a. Go to **File…Import**.



b. From the resulting dialogue box, expand **General** if necessary, and then select **Existing Projects into Workspace**. Hit **Next.**



Select

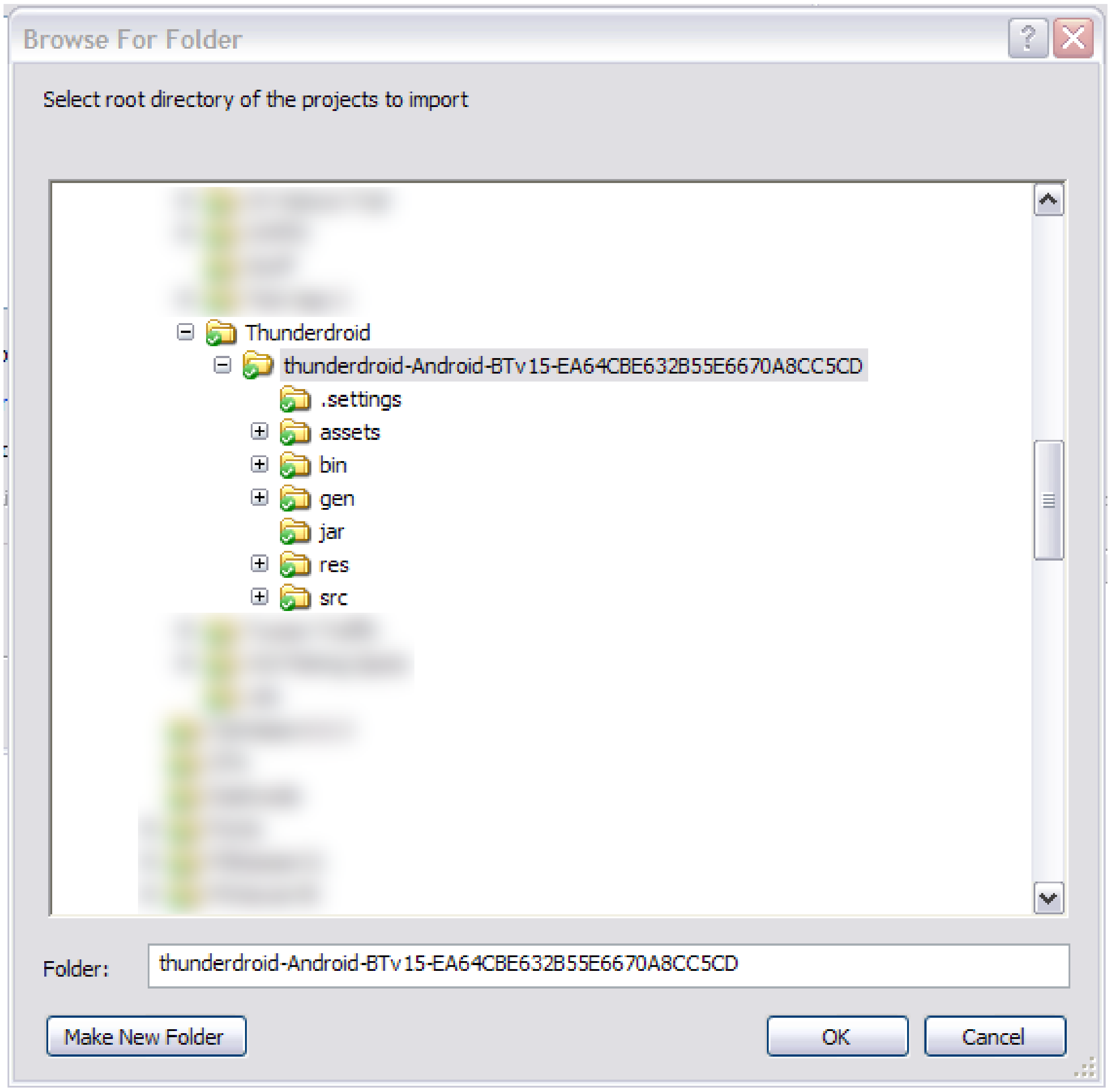
Create new projects from an archive file or directory.

Select an import source:

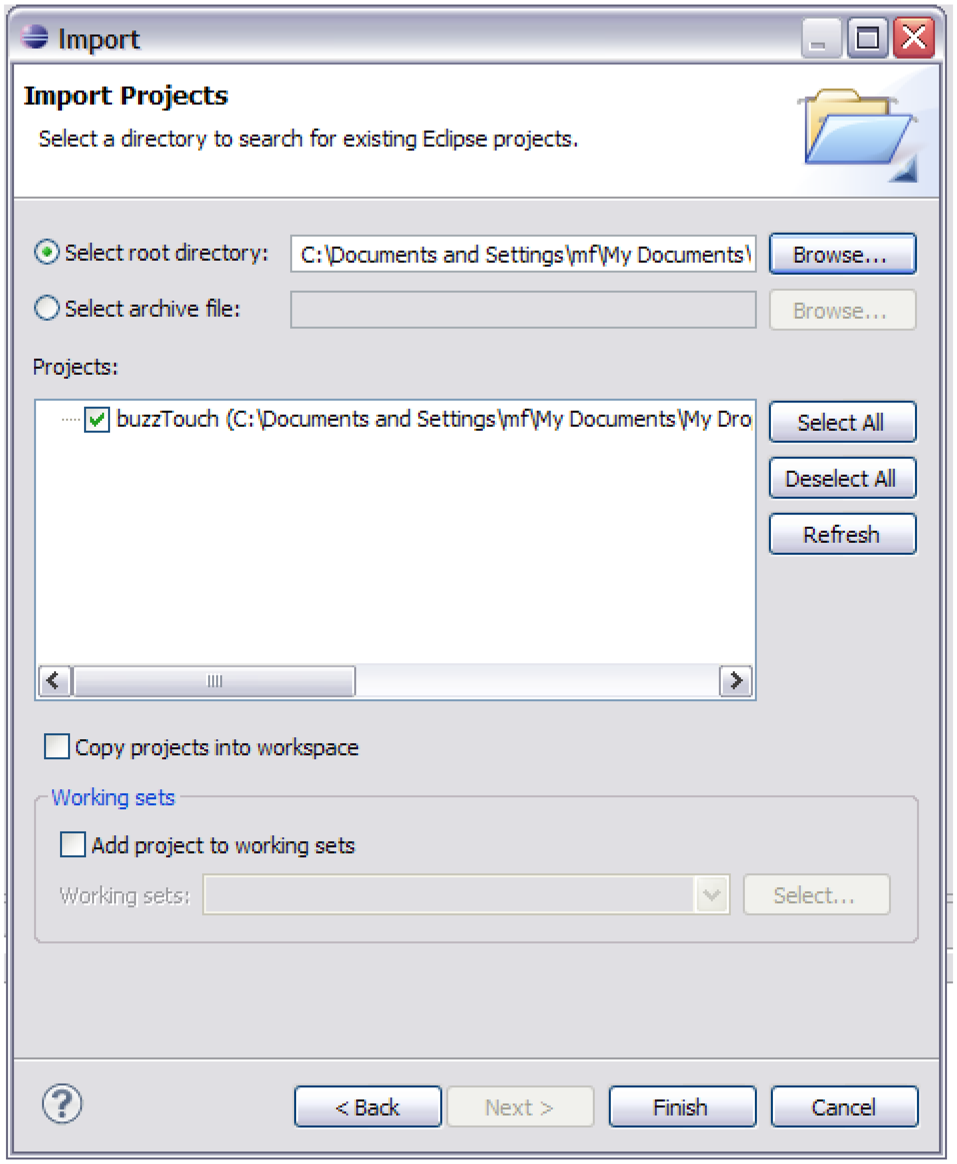
type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- CVS
- Install
- Plug-in Development
- Run/Debug
- Tasks
- Team
- XML

c. Make sure **Select root directory** is selected, then hit **Browse** and select the unzipped folder that you got from buzztouch. Just select the root folder…none of the ones beneath it, and none of the files individually.


Browse For Folder
Select root directory of the projects to import
Thunderdroid
thunderdroid-Android-BTv15-EA64CBE632B55E6670A8CC5CD
.settings
assets
bin
gen
jar
res
src
Folder:
thunderdroid-Android-BTv15-EA64CBE632B55E6670A8CC5CD
Make New Folder
OK
Cancel

d. Your project should show up in the **Projects:** box, and should be check marked. Hit **Finish**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\Documents and Settings\mf\My Documents\

○ Select archive file:

Projects:

☑ buzzTouch (C:\Documents and Settings\mf\My Documents\My Dro

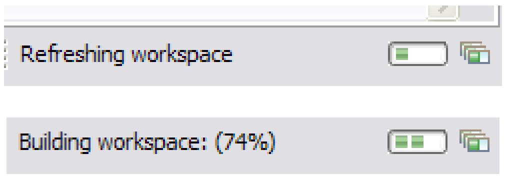
☐ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

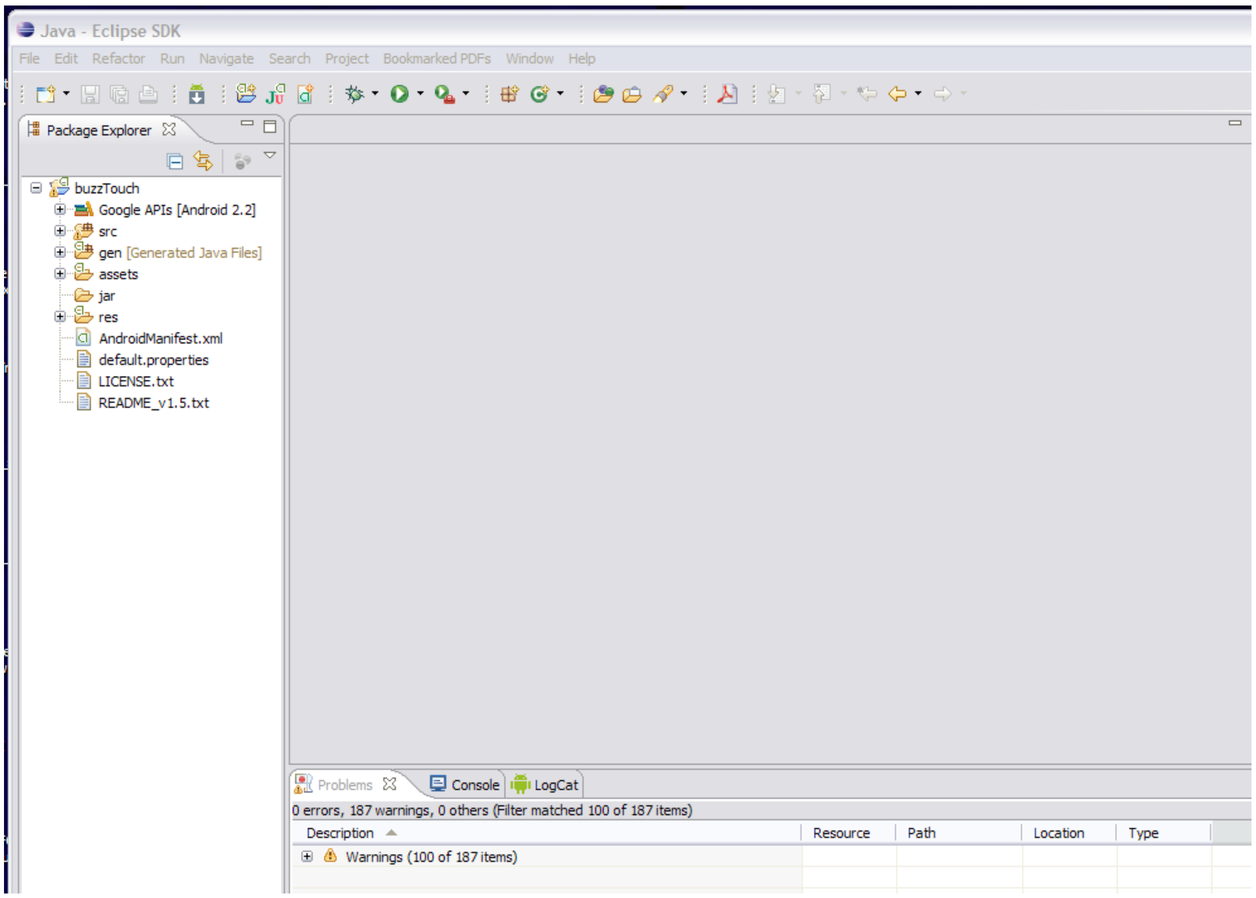
e. Wait while Eclipse builds your workspace. You can see the progress in the lower right-hand corner of Eclipse.



Refreshing workspace

Building workspace: (74%)

f. Once the workspace is built, your project should be loaded and ready for you to work with. You should see a screen similar to the following. We'll address the warnings and stuff later…this is just about getting your project into Eclipse.



0 errors, 187 warnings, 0 others (Filter matched 100 of 187 items)

Warnings (100 of 187 items)

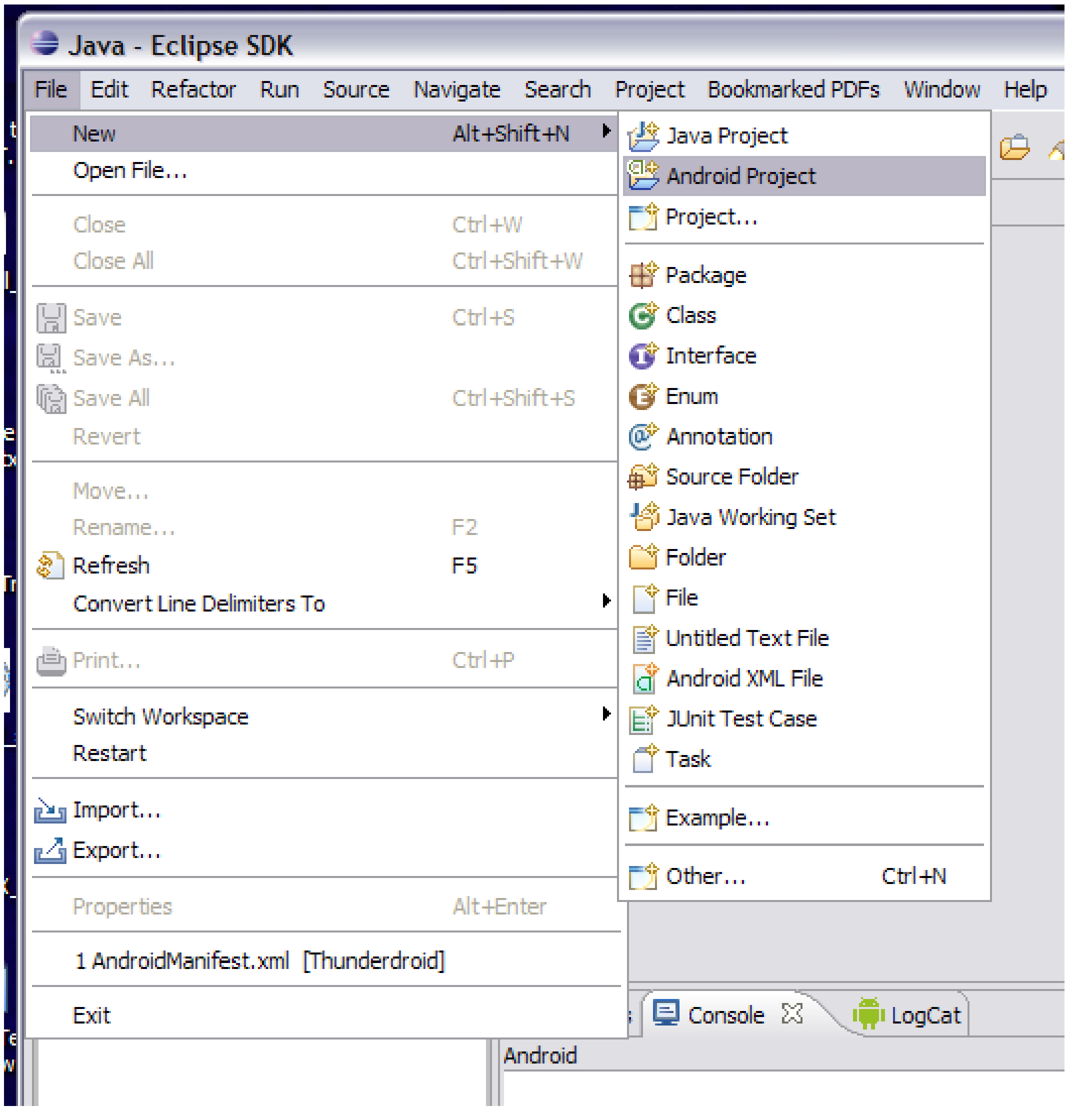
Method Two: Start A New Android Project.

This method starts an entirely new project, but uses files that you point Eclipse to. It'll say a project already exists in your source folder, but that's OK, just select it, and it'll build the project based on those files.

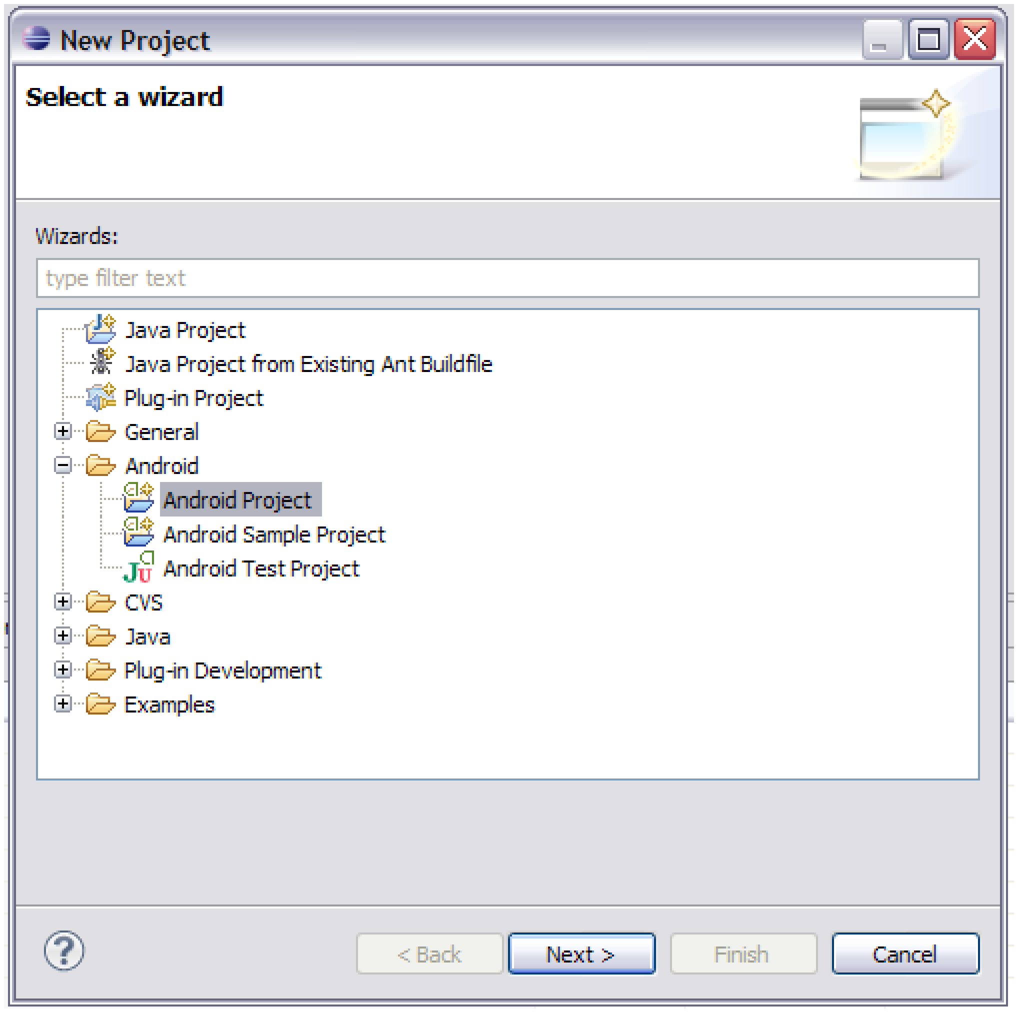
1. **Start Eclipse**. Start Eclipse by either clicking on the icon you set up for it somewhere, or double-clicking on the eclipse.exe file that is found In your installation path. For me, it's at c:\eclipse\eclipse.exe (Windows).

2. **Start a new Android Project.**

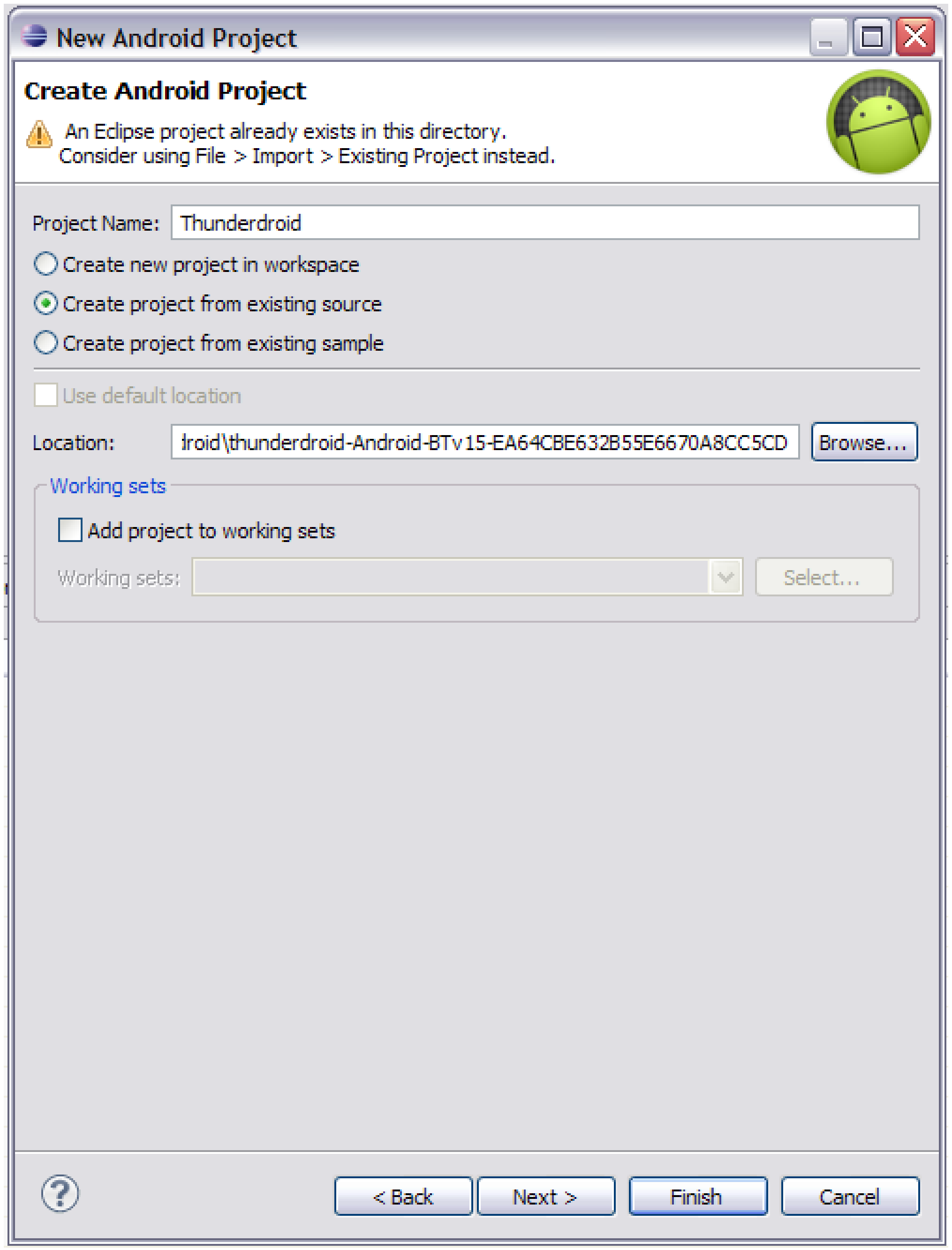
    1. Go to **File…New…Project**. You might have an option for Android Project, but you might not. So, just select Project



2. From the resulting dialogue box, expand **Android** if necessary, and select **Android Project.** Hit **Next.**



3. Specify a **Project Name** (I use the same name as what I have in the BT control panel for the app). Select **Create Project From Existing Source** and browse to your download folder. Select the root folder from the source package you downloaded and unzipped from BT. Hit **Finish.**


New Android Project

Create Android Project

An Eclipse project already exists in this directory.
Consider using File > Import > Existing Project instead.

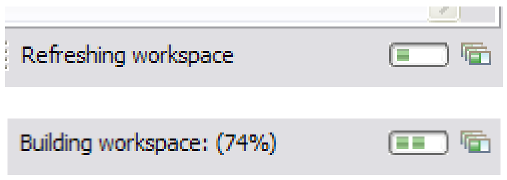
Project Name: Thunderdroid

( ) Create new project in workspace
[✓ (•) Create project from existing source]
( ) Create project from existing sample

[ ] Use default location

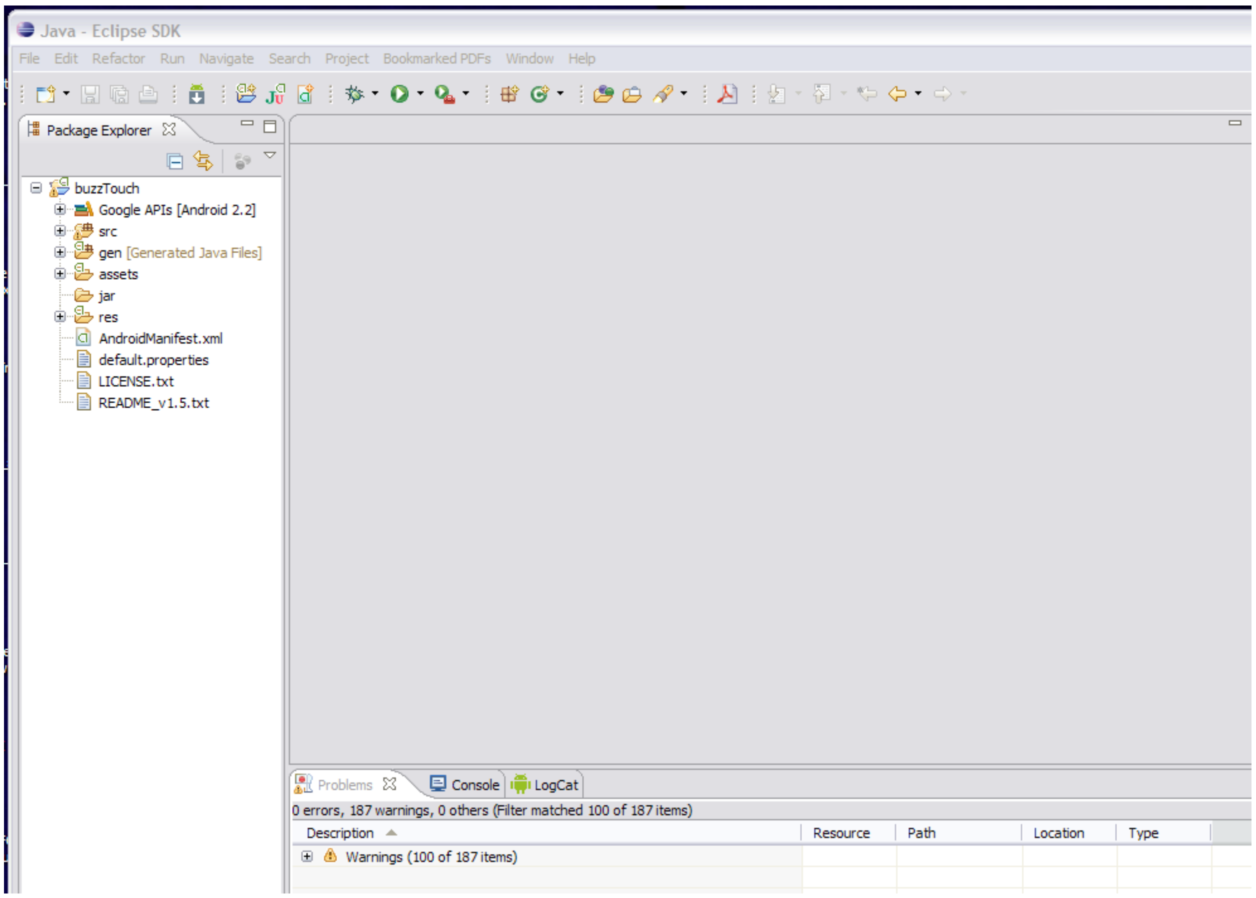
Location: droid\thunderdroid-Android-BTv15-EA64CBE632B55E6670A8CC5CD

4. Wait while Eclipse builds your workspace. You can see the progress in the lower right-hand corner of Eclipse.


Refreshing workspace

Building workspace: (74%)

5. Once the workspace is built, your project should be loaded and ready for you to work with. You should see a screen similar to the following. We'll address the warnings and stuff later…this is just about getting your project into Eclipse.

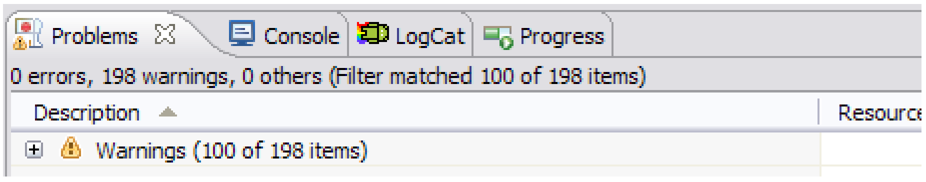


0 errors, 187 warnings, 0 others (Filter matched 100 of 187 items)

Warnings (100 of 187 items)

Congratulations! Your project is now loaded into Eclipse and ready for you to work with. See…that wasn't so bad!

Before you can actually run your application, you'll need to make sure it's **error free**. It's OK if there are warnings in your application…those are just what they sound like, but you can't have any errors. Take a look at the **Problems** tab and see what's there.

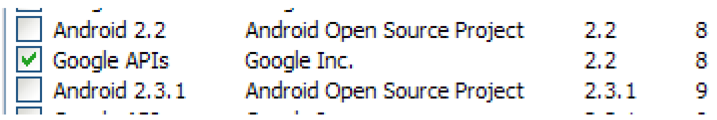


0 errors, 198 warnings, 0 others (Filter matched 100 of 198 items)

Warnings (100 of 198 items)

In this particular example, there are only warnings, so that's a good thing. However…initially, when I loaded this project, it came up with a bunch of errors. I struggled for a bit to understand why, and then decided to check and make sure the correct Google API was being referenced. It was not, and when I corrected it, all was well. So, if you see errors in you project, do the following:

1. Go to **File….Properties** and click on **Android.** For buzztouch v1.5/v2.0 apps, you need to make sure you have selected **Google APIs 2.2/8**. If you have not, none of this will work. Make sure that, and only that, API is checked, and hit **Apply**. Then hit **OK.**



☐	Android 2.2	Android Open Source Project	2.2	8
✓	Google APIs	Google Inc.	2.2	8
☐	Android 2.3.1	Android Open Source Project	2.3.1	9

Go to **File…Refresh (F5)** and refresh your project. Hopefully all the errors will disappear! If they don't, something else is wrong, and you'll need to post a message on the forum to ask for help, or see the list of Common Android Issues later in this book.

Now that your app is error free (remember, warnings are OK), you're ready to compile and run it in the simulator. You should have an Android Virtual Device set up in Eclipse per my other book. It will get loaded automatically when you run your app.

To run your app in the virtual device, do the following:

1. Highlight your project in Package Explorer

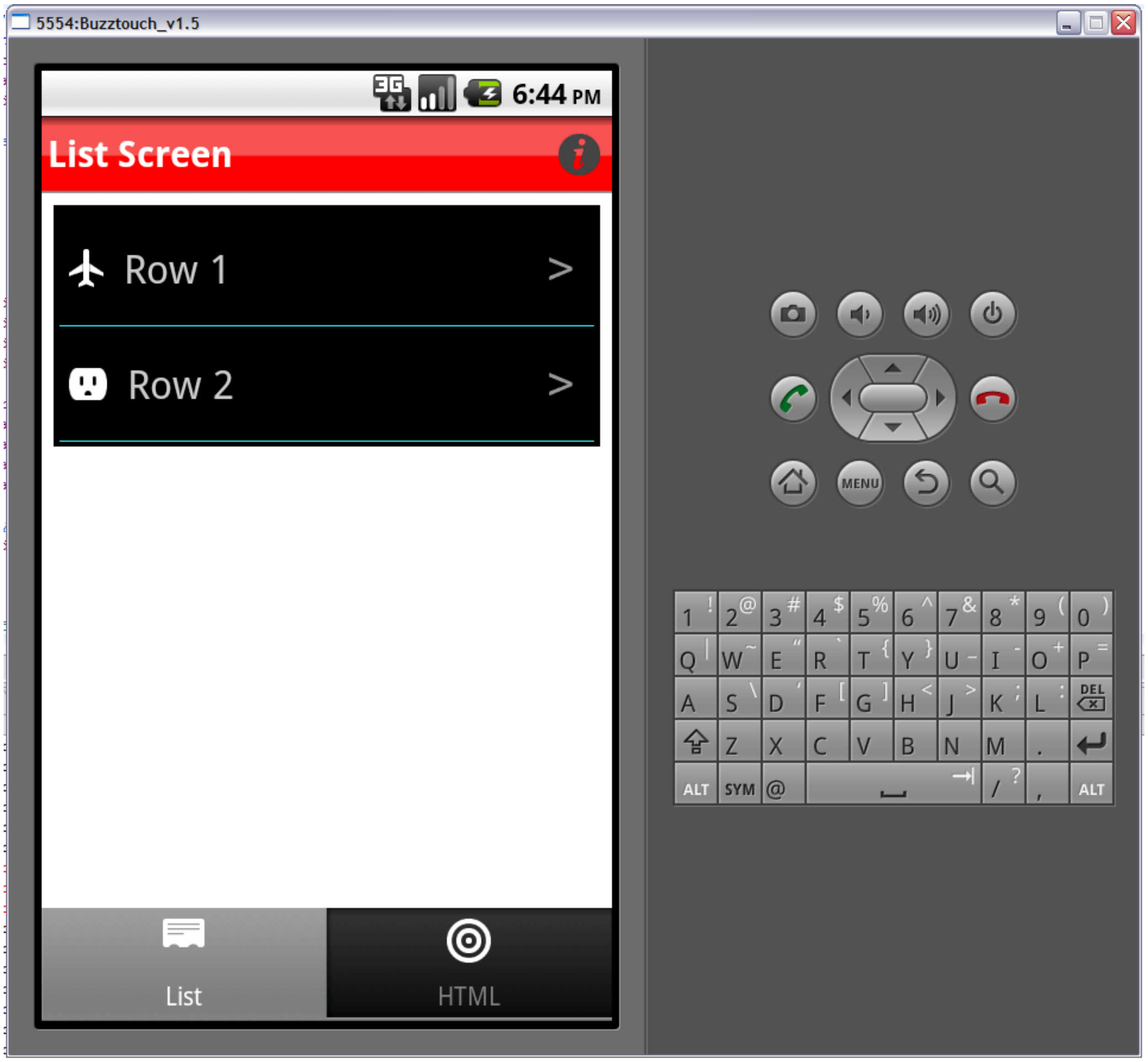
2. Go to **Run…Run As…Android Application.** Or, you could hit the green circle icon you see on the toolbar.

3. Click on the **Console** tab so you can monitor what's happening. If things are going well, you'll see output that looks like this:

```
[2011-11-30 14:26:28 - buzztouch] ------------------------------
[2011-11-30 14:26:28 - buzztouch] Android Launch!
[2011-11-30 14:26:28 - buzztouch] adb is running normally.
[2011-11-30 14:26:28 - buzztouch] Performing com.thunderdroid.BT_activity_root activity
launch
[2011-11-30 14:26:29 - buzztouch] Automatic Target Mode: launching new emulator with
compatible AVD 'buzztouch_v1.5'
[2011-11-30 14:26:29 - buzztouch] Launching a new emulator with Virtual Device
'buzztouch_v1.5'
[2011-11-30 14:27:15 - buzztouch] New emulator found: emulator-5554
[2011-11-30 14:27:15 - buzztouch] Waiting for HOME ('android.process.acore') to be
launched...
```

4. Wait while everything loads…this often takes several minutes. Seriously…you can get a cup of coffee, or perhaps arrange to send me a case of SoBe Fruit Punch (my favorite drink!) while it's loading. The AVD is super slow to load.

If everything goes well, your app will show up in your AVD, and you'll be doing the happy dance!



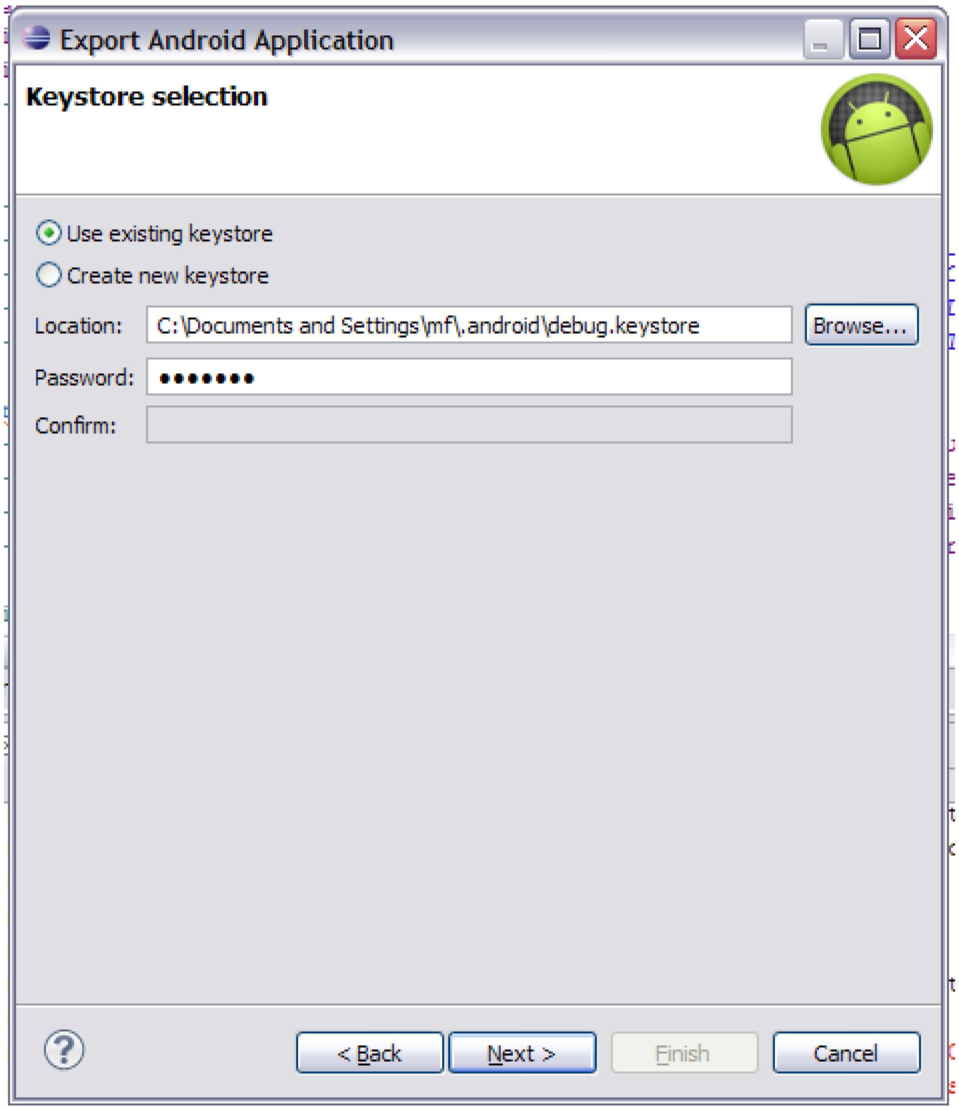
(Yeah, I know that's a totally lame looking app, but it was just something I threw together to demonstrate this stuff for you. I'm working on some better stuff…I promise!)

So, now you know that your app works, and after spending countless hours making it look beautiful, you're ready to test it on a real device, or publish it somewhere for the general masses. Let's take a look at how you **Export** your app and create the **.apk** package you need in order to install it on a device.

To start, we're only going to worry about creating the debug package, using the debug.keystore, until we have a more complete discussion about keystores in a future chapter.

Here's the process for creating a **debug package**:

1. Highlight your project in Package Explorer.
2. Go to **File…Export**, and select **Export Android Application**. Expand the Android section if you don't see that option. Hit **Next.**
3. Select the project to export. It should already be filled in with your project name (most likely *buzztouch*). If not, browse to the folder your project is in. Note the statement on the screen that says "The manifest 'debuggable' attribute is set to true." This is what you want. Hit **Next**.
4. Navigate to your **debug.keystore** file, and enter the keystore password ('**android**' is the default). Hit **Next.**

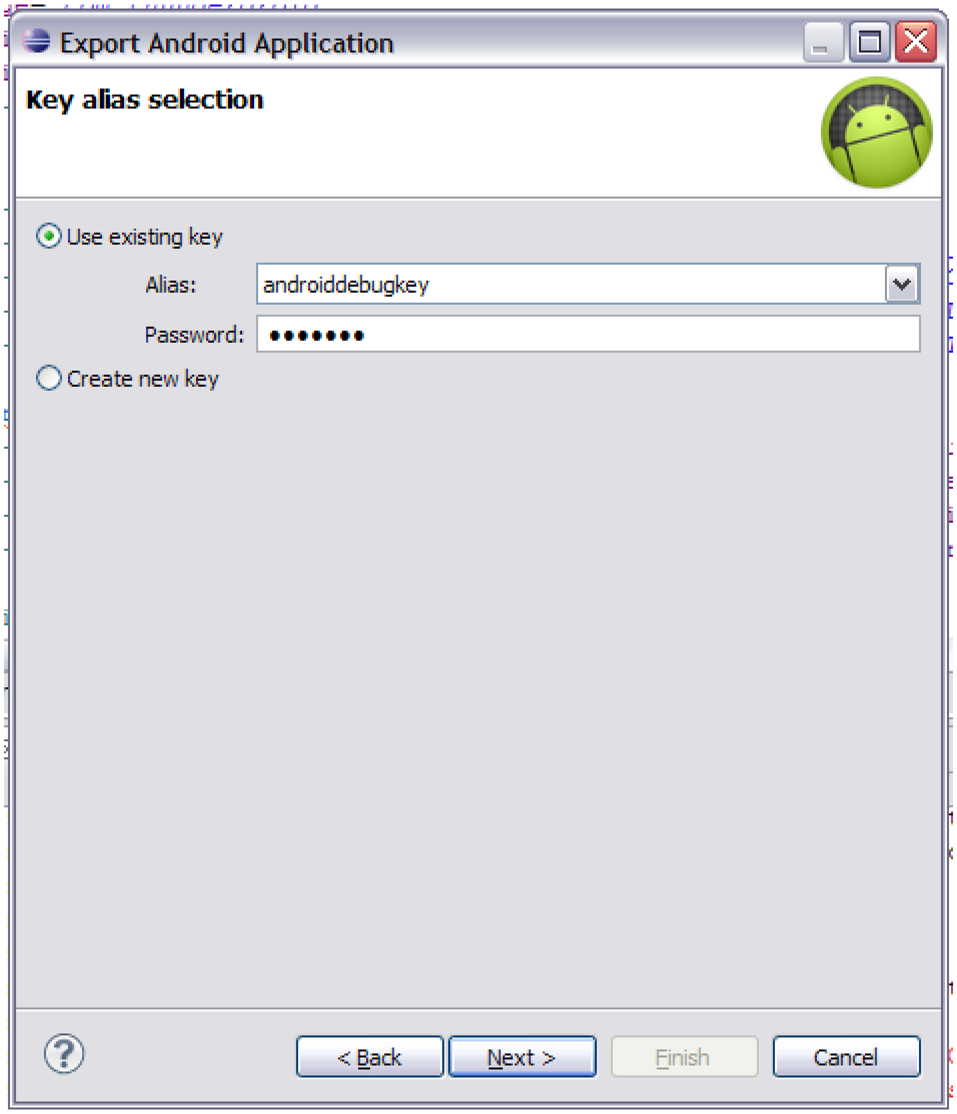

Export Android Application

Keystore selection

[✓ Use existing keystore]
Create new keystore

Location: C:\Documents and Settings\mf\.android\debug.keystore
Password: •••••••
Confirm:

5. Choose **Use Existing Key**, and select your debug.keystore alias (default is 'androiddebugkey') and password ('android'). Hit **Next**.



Key alias selection

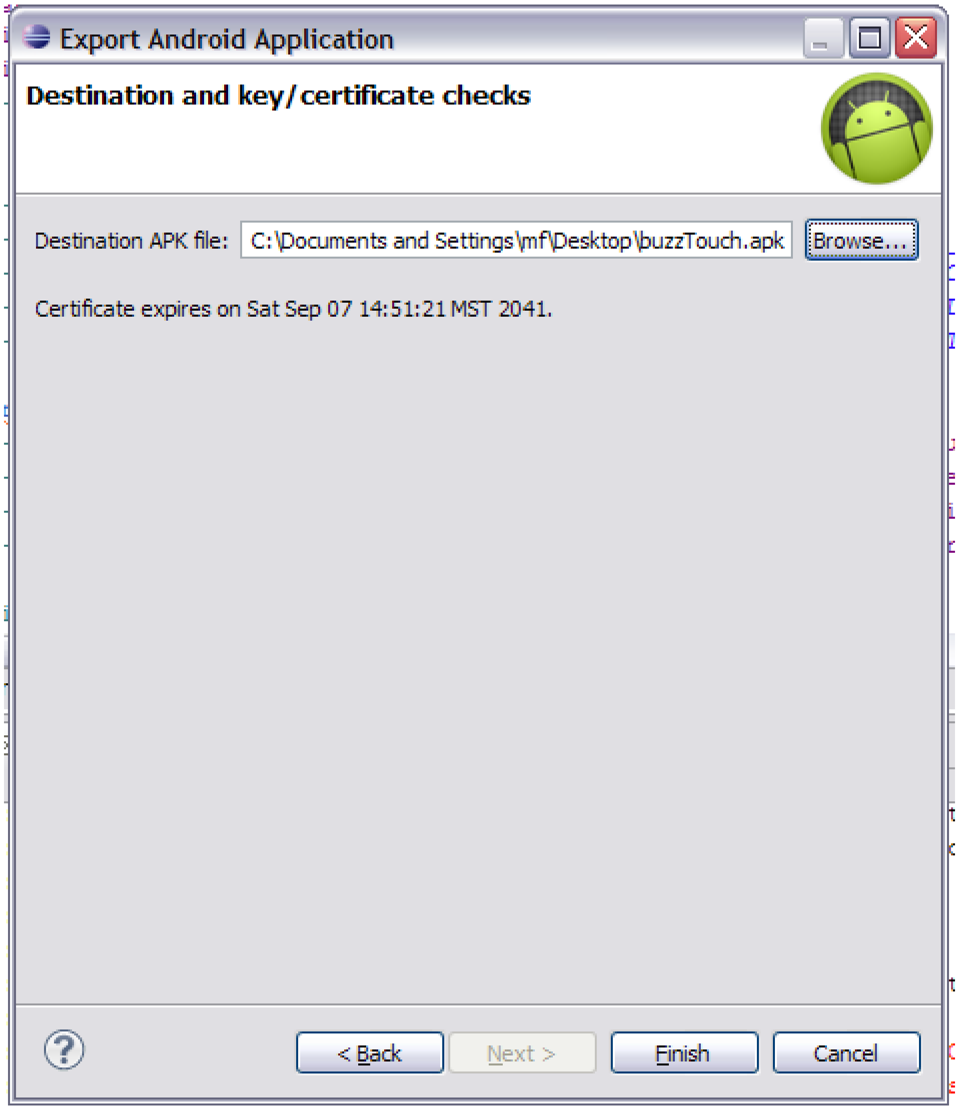
[✓ Use existing key]

Alias: androiddebugkey

Password: ●●●●●●●

Create new key

6. **Browse** to the location you want to have your package saved to, then hit **Finish.**

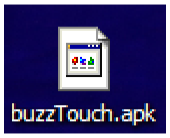


Destination and key/certificate checks

Destination APK file: C:\Documents and Settings\mf\Desktop\buzzTouch.apk

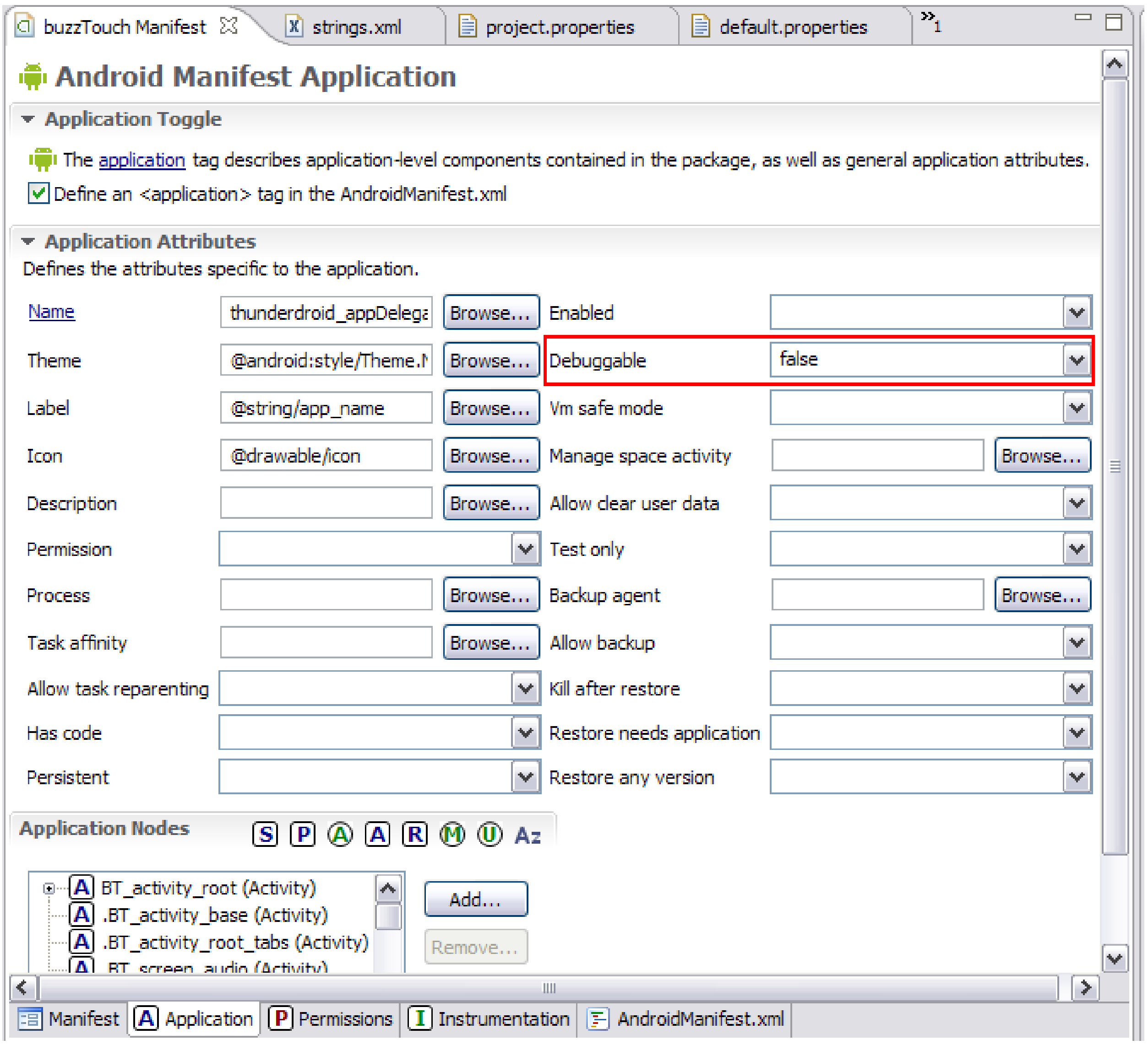
Certificate expires on Sat Sep 07 14:51:21 MST 2041.

7. Check the location to make sure the file was created! If all is good, you can now deploy this on your test device. I've found that putting it on the web somewhere and having a person surf to the package is the easiest way to install.



Creating a **release package** is basically the same process as a debug package. The main differences are the keystore you use, and the fact that the debuggable attribute is set to 'false.' Here's what you do:

1. Double-click on the **AndroidManifest.xml** file in your project tree. You'll find that in Project Explorer. Several tabs should open up, one called **Application**. In that tab you will find many fields that can be changed, one of which is called **Debuggable**. You want to set this to **false.**



Android Manifest Application

Application Toggle

The application tag describes application-level components contained in the package, as well as general application attributes.

☑ Define an <application> tag in the AndroidManifest.xml

Application Attributes

Defines the attributes specific to the application.

Name	thunderdroid_appDelega	Enabled	
Theme	@android:style/Theme.N	Debuggable	false
Label	@string/app_name	Vm safe mode	
Icon	@drawable/icon	Manage space activity	
Description		Allow clear user data	
Permission		Test only	
Process		Backup agent	
Task affinity		Allow backup	
Allow task reparenting		Kill after restore	
Has code		Restore needs application	
Persistent		Restore any version	

Application Nodes

- BT_activity_root (Activity)
- .BT_activity_base (Activity)
- .BT_activity_root_tabs (Activity)
- .BT_screen_audio (Activity)

2. Highlight your project in Package Explorer.

3. Go to **File…Export**, and select **Export Android Application**. Expand the Android section if you don't see that option. Hit **Next.**

4. Select the project to export. It should already be filled in with your project name. If not, browse to the folder your project is in. Hit **Next**.

5. Navigate to your **release.keystore** file, and enter the keystore password (I have no idea what it would be for you). Hit **Next.**

6. Choose **Use Existing Key**, and select your **release.keystore** alias and password. Hit **Next**.
7. **Browse** to the location you want to have your package saved to, then hit **Finish.**

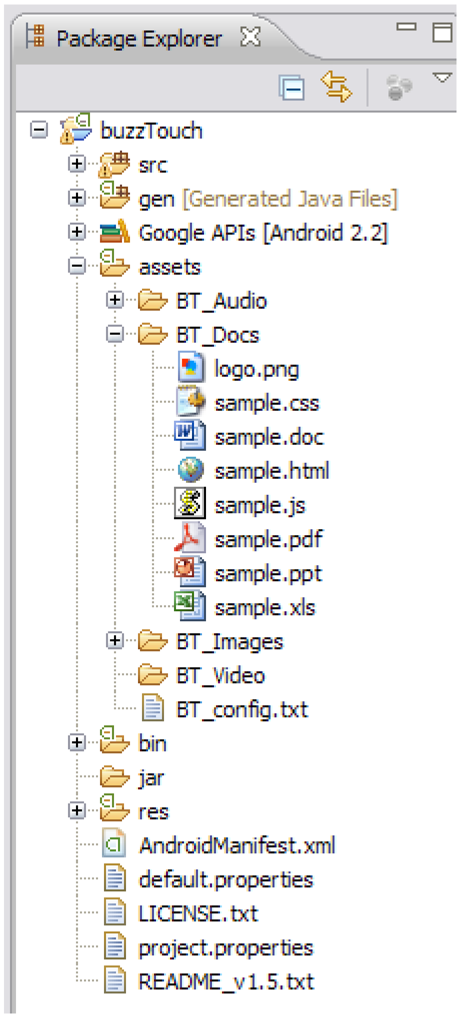
That’s it! Just like with the debug process, you should now have a .apk file somewhere that can be loaded into whatever marketplace you want to distribute from.

Chapter 8

Adding Files to Your Project

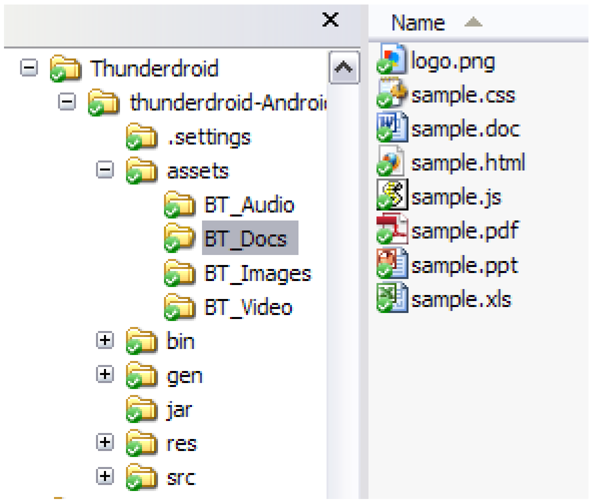
There may come a time when you want to include files in the app itself, which means the content is immediately available to your users (instead of downloading via the network). Such content might include images, books (HTML, Word, Excel, PDF), audio and video. Most buzztouch projects handle this without any problems, as you are able to specify both local and URL files. If you specify a local one, however, it needs to be added to your project files so it can be compiled into your app. Here's how you do that!

1. Open your project in Eclipse, and view it in Package Explorer. That's window you see on the left side of the screen. This will show you the file structure for your project. Notice that this project has the default files provided by buzztouch.

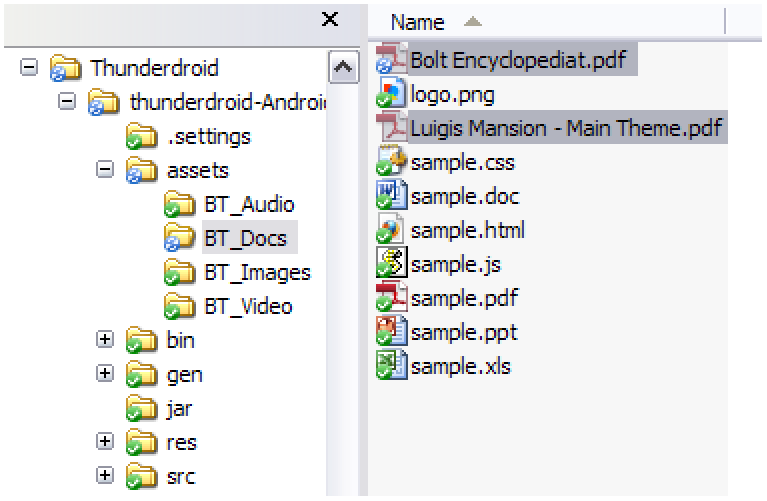


```
buzzTouch
    src
    gen [Generated Java Files]
    Google APIs [Android 2.2]
    assets
        BT_Audio
        BT_Docs
            logo.png
            sample.css
            sample.doc
            sample.html
            sample.js
            sample.pdf
            sample.ppt
            sample.xls
        BT_Images
        BT_Video
        BT_config.txt
    bin
    jar
    res
    AndroidManifest.xml
    default.properties
    LICENSE.txt
    project.properties
    README_v1.5.txt
```

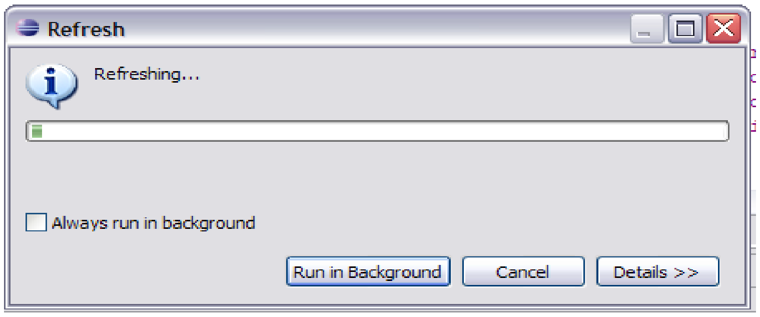
2. Open your buzztouch project in Windows Explorer or Mac Finder, and navigate to the BT_Docs folder. Notice that what you see here, and what you see in Package Explorer, they match.



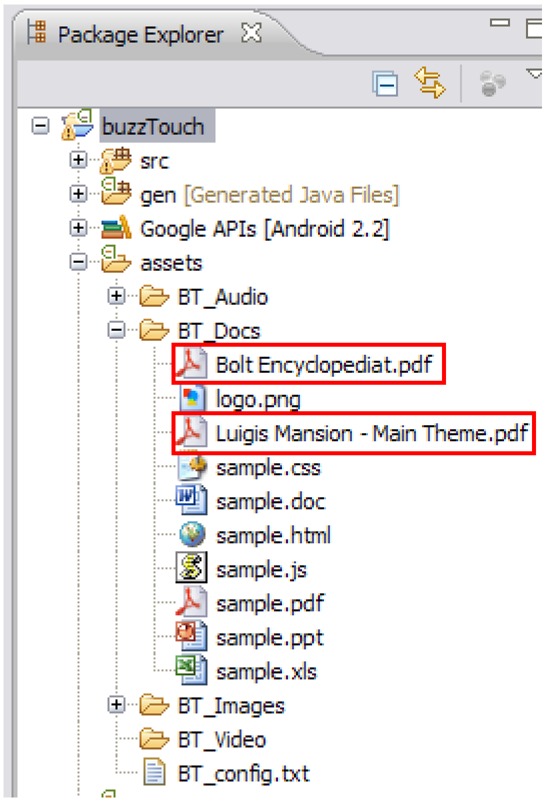
3. Add the files you want to be included to your BT_Docs folder in Explorer or Finder.



4. Go to **File…Refresh**, or press the **F5 Key**, and the project will begin to refresh. The following dialogue box will pop up.



5. Verify that your files are now visible In the Package Explorer.



6. Do a **Project…Clean** and then **Run…Run As…Android Application**. Your files should now be part of your app!

Overall the process is pretty simple. The key thing to always keep in mind is that new files added to your project folder…the set of folders that you downloaded from buzztouch…must also be added to both your Xcode and Eclipse projects. If you don't indicate that there are new files that should be included in the build process, then they'll never show up in your app. Follow these steps, and you should be OK!

Chapter 9

All About Keystores and Signing

If you are creating an Android application, Google requires that those applications are signed with a certificate. This signing process is significantly easier than what Apple makes iOS developers go through, as Android developers can generate their own certificates for signing applications, using commonly available tools like *keystore*, which comes with your Java JDK. Google requires that **all** applications be signed, be it a development or release application. Each requires a different certificate, so let's take a look at each.

Signing For Development/Debug

When you installed your Android SDK, you got with it a debug.keystore that can be used to sign your application for debug purposes. The default locations for the debug.keystore appears to be these locations for the following platforms:

Windows: C:\Books and Settings\<USER>\.android

Mac: \Users\<USER>\android

If you don't find anything at these locations on your computer, just do a search for 'keystore' and you should find what you're looking for.

The debug.keystore ships with the following default parameters:

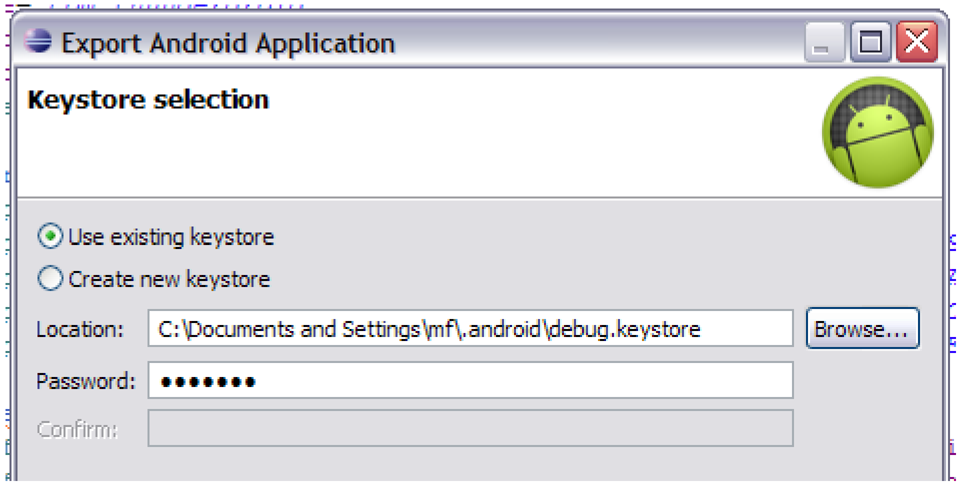
- Keystore name: "debug.keystore"
- Keystore password: "android"
- Key alias: "androiddebugkey"
- Key password: "android"
- CN: "CN=Android Debug,O=Android,C=US"

This information becomes important when you go to sign your application for debug. Once you have your application ready for testing, and you want to create a debug package, you can follow these steps:

1. Make sure that **android:debuggable="true"** is set in your AndroidManifest.xml file. This file is located in the root of your project folder that you got from buzztouch.
2. Highlight your project in Package Explorer, right-click, and select **Export**.
3. Select **Android…Export Android Application** and then **Next.**

4. The Project name should already be filled in, since you are working directly from that project. Hit **Next.**

5. Select **Use Existing Keystore** and hit Browse to navigate to your debug.keystore, and select it. Then enter your password (android), and hit **Next**.



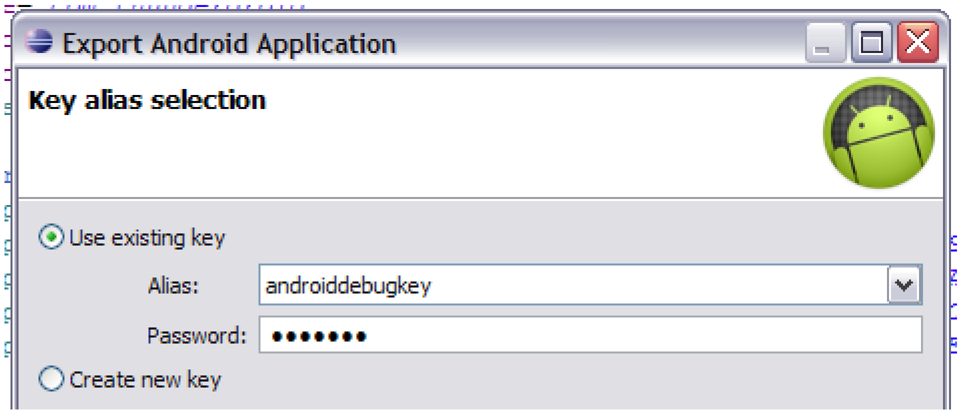
(● Use existing keystore) ○ Create new keystore

Location: C:\Documents and Settings\mf\.android\debug.keystore

Password: •••••••

Confirm:

6. Select the **androiddebugkey** alias and use **android** as the password. Select **Next.**



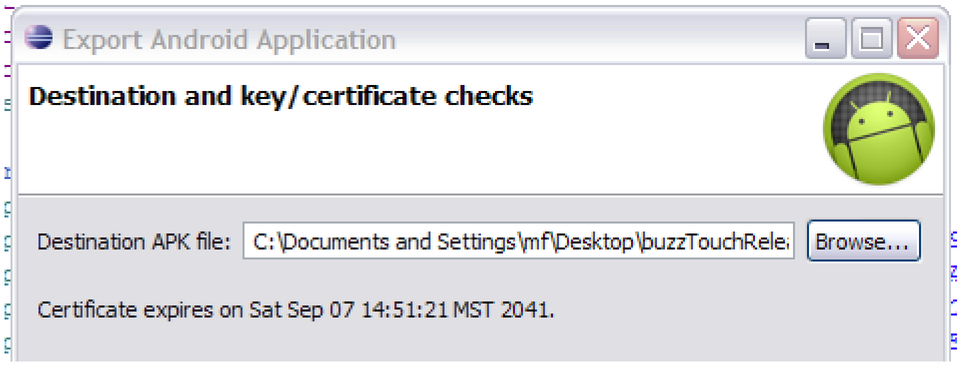
● Use existing key

Alias: androiddebugkey

Password: •••••••

○ Create new key

7. Now you'll want to select a destination for your .apk file. As always, make sure it's somewhere you remember. You can also give your package a name at this step. Notice that in this example, the expiration date for the debug key is in 2041. So no worries!


Export Android Application

Destination and key/certificate checks

Destination APK file: C:\Documents and Settings\mf\Desktop\buzzTouchRele

Certificate expires on Sat Sep 07 14:51:21 MST 2041.

8. Hit **Finish**, and the debug package for your application will be built! Congratulations on building and signing your Android application! You can now distribute it in whichever way you want (email, the web, etc), so long as it doesn't go into a store. Debug apps are not allowed in stores (meaning, if debuggable=”true” is set, it won't get accepted).

Keep in mind that you don't have to use the default debug.keystore that comes with your Android SDK. I'm not entirely sure why you would want to create your own…perhaps you want people to specifically know it came from you, but the option exists to create it during this same process. Just select **Create New Keystore** instead of using the existing keystore, and you'll be prompted through the process.

Signing For Release/Distribution

OK, you've done all your testing, everything is in order, and you're ready to put your store in one of the various marketplaces. Cool! The process for signing your application for release/distribution is the same as for debug/development, but in this case you use a release.keystore instead. You can create a release.keystore during the export process.

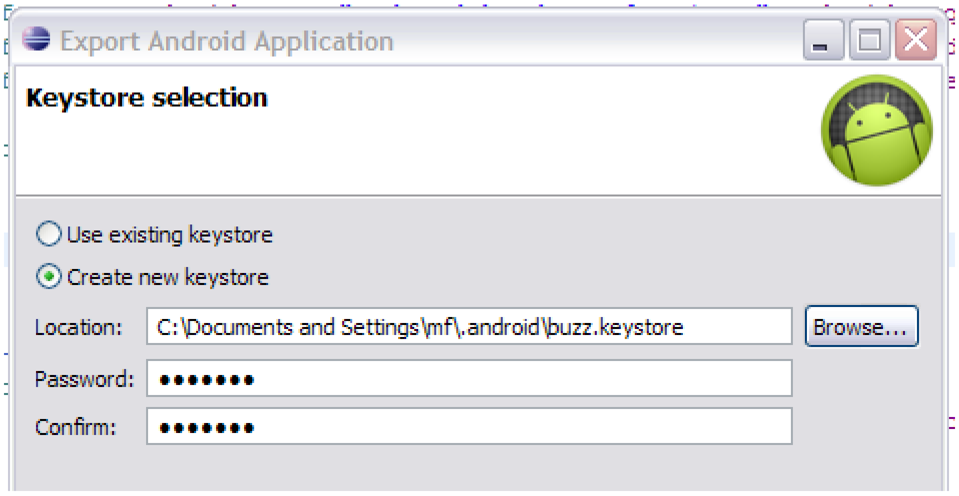
1. Make sure that **android:debuggable="false"** is set in your AndroidManifest.xml file. This file is located in the root of your project folder that you got from buzztouch.

2. Highlight your project in Package Explorer, right-click, and select **Export**.

3. Select **Android…Export Android Application** and then **Next.**

4. The Project name should already be filled in, since you are working directly from that project. Hit **Next.**

5. Select **Create new keystore** and fill in a keystore location and passwords (keep them strong!).

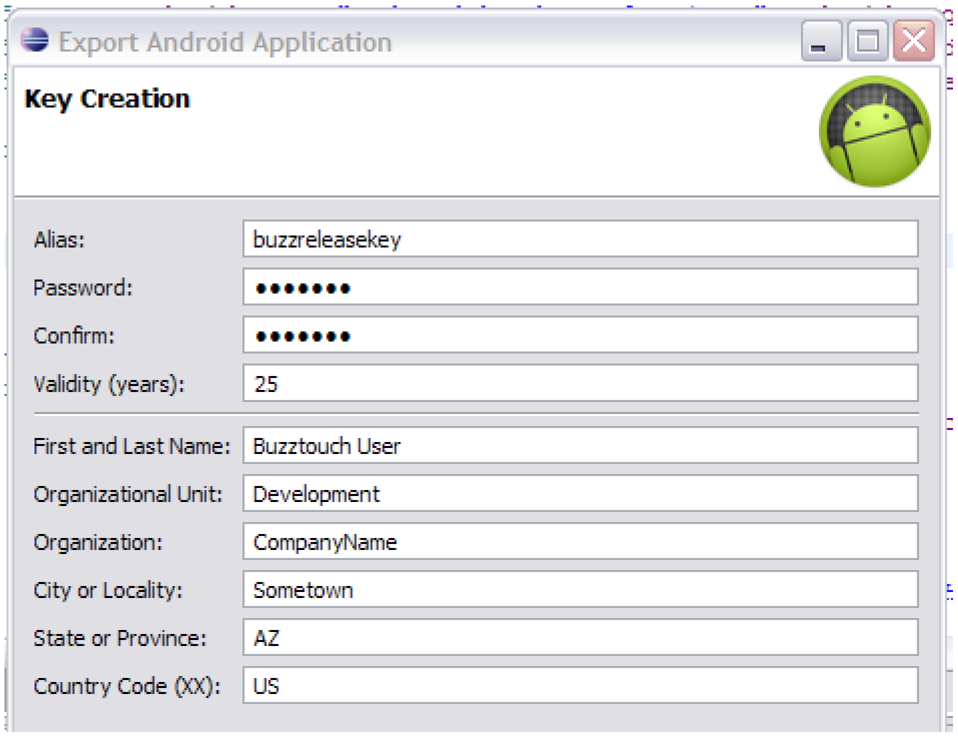


Keystore selection

( ) Use existing keystore
(●) Create new keystore

Location:	C:\Documents and Settings\mf\.android\buzz.keystore
Password:	•••••••
Confirm:	•••••••

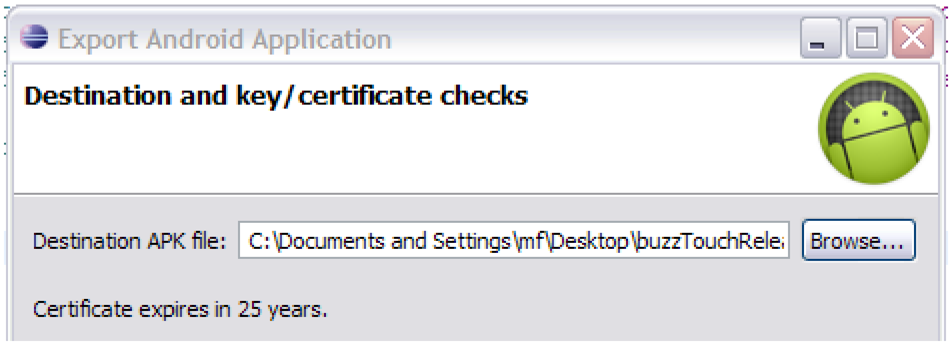
6. Provide the requested information. Use a strong password, and pick a nice validity duration (I used 25 years). This can be the release key you use forever, and it identifies you as the signer of the application. So, keep it legit, and keep it secure! Hit **Next.**



Key Creation

Alias:	buzzreleasekey
Password:	•••••••
Confirm:	•••••••
Validity (years):	25
First and Last Name:	Buzztouch User
Organizational Unit:	Development
Organization:	CompanyName
City or Locality:	Sometown
State or Province:	AZ
Country Code (XX):	US

7. Now you'll want to select a destination for your .apk file. As always, make sure it's somewhere you remember. You can also give your package a name at this step.



Destination and key/certificate checks

Destination APK file: C:\Documents and Settings\mf\Desktop\buzzTouchRele

Certificate expires in 25 years.

8. Hit **Finish**, and the release package for your application will be built! Congratulations on building and signing your Android application! You are now ready to officially place it in the Marketplace!

That pretty much sums up the Android signing process! The two keystores you created, debug and release, will be used again in a different tutorial on getting a Google Maps API key. You should make a backup of our release key, as it identifies you as the author of your app, and should not be compromised.

Chapter 10

Getting a Google Maps API Key

If you are creating an Android application that uses maps in it (which is a very cool feature to have!), you're going to need a Google Maps API key from, you guessed it, Google! The process can be a bit confusing, and is booked here:

http://code.google.com/android/add-ons/google-apis/mapkey.html

Let's take a few minutes and walk through the process step-by-step. We'll do this on a Windows XP box, but the steps are essentially the same on a Mac or Linux box. You're going to do the following:

- Get the MD5 Fingerprint of your Signing Certificate
- Get the MD5 Fingerprint of your SDK Debug Certificate
- Register your fingerprints (not the ones on your hand!) with the Google Maps Service
- Add the Maps API Keys to your application

Let's begin by making sure the appropriate keystores are in place. When you installed your Android SDK, it automatically created a debug.keystore, which you use for, you guessed it, debug signing! The default locations for the debug.keystore appear to be these locations for the following platforms:

Windows: C:\Books and Settings\<USER>\.android

Mac: \Users\<USER>\android

If you don't find anything at these locations on your computer, just do a search for 'keystore' and you should find what them.

The debug.keystore ships with the following default parameters:

- Keystore name: "debug.keystore"
- Keystore password: "android"
- Key alias: "androiddebugkey"
- Key password: "android"

CN: "CN=Android Debug,O=Android,C=US"

If, for some reason, you debug.keystore does not exist, or become corrupted, you can create a new one by doing a build of your application, and it will automatically be created.

In addition to the debug.keystore, you'll also need a release keystore created in order to get your Google Maps API key. There are basically two ways to create it – via the command line, and by building your application for release, and creating the keystore at that time.

1. **Command Line Method.** Open a command prompt (Windows)or terminal session (Linux/Mac) and issue the following command:

    a.
    ```
    keytool -genkey -v -keystore you-decide.keystore -alias
    alias_name -keyalg RSA -keysize 2048 -validity 10000
    ```

2. **Release Build Method.** With this method, you will build your app in release mode, and create the keystore along the way.

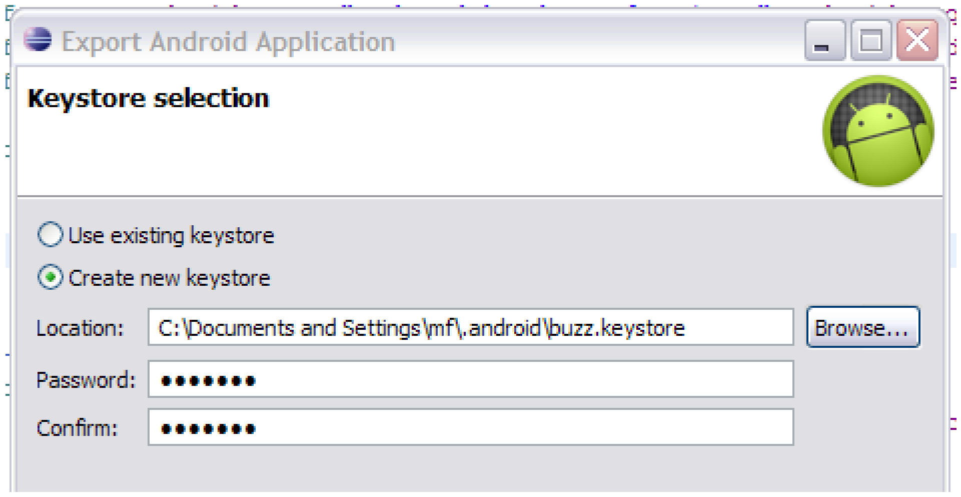
    a. Make sure that **android:debuggable="false"** is set in your AndroidManifest.xml file. This file is located in the root of your project folder that you got from buzztouch.

    b. Highlight your project in Package Explorer, right-click, and select **Export**.

    c. Select **Android…Export Android Application** and then **Next.**

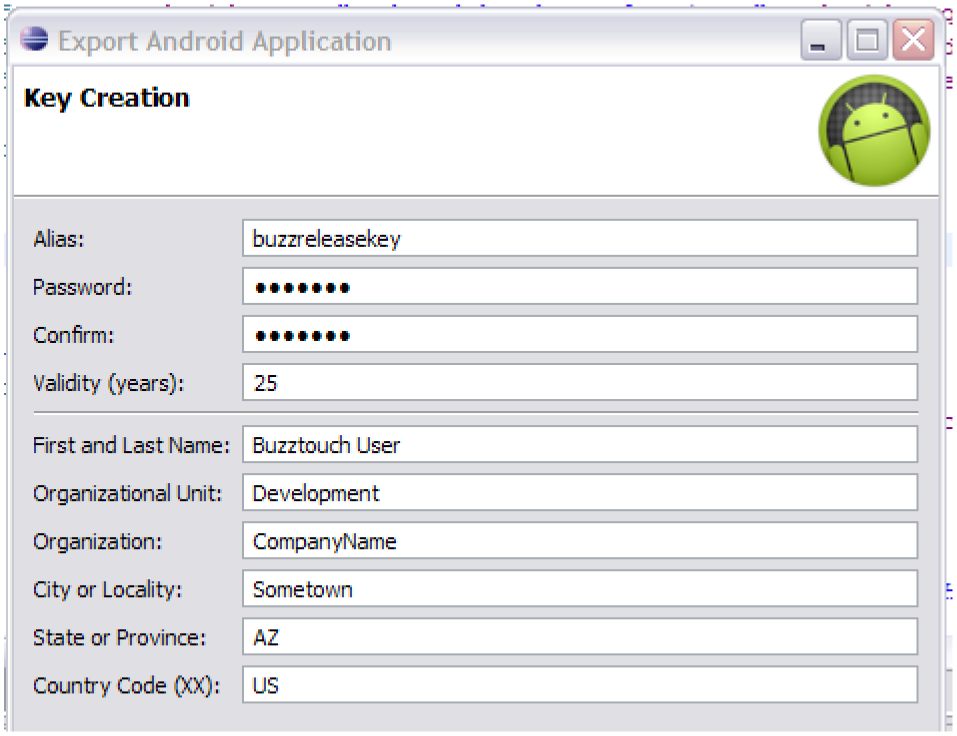
    d. The Project name should already be filled in, since you are working directly from that project. Hit **Next.**

    e. Select **Create new keystore** and fill in a keystore location and passwords (keep them strong!).



[ ] Use existing keystore
[✓] Create new keystore
Location: C:\Documents and Settings\mf\.android\buzz.keystore
Password: •••••••
Confirm: •••••••

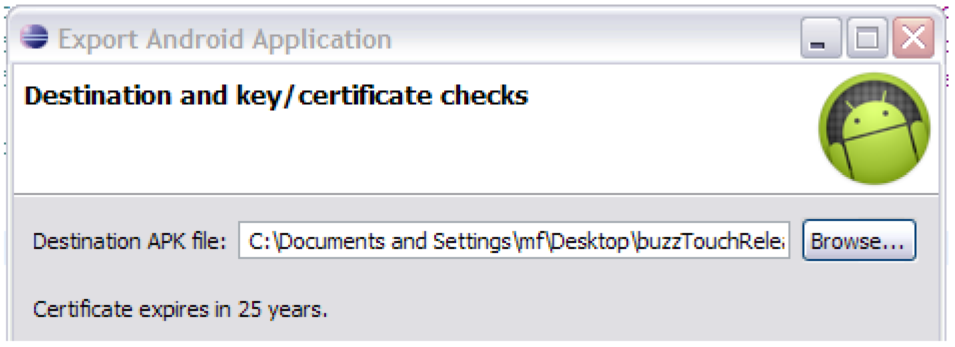
    f. Provide the requested information. Use a strong password, and pick a nice validity duration (I used 25 years). This can be the release key you use forever, and it identifies you as the signer of the application. So, keep it legit, and keep it secure! Hit **Next.**


Export Android Application

Key Creation

Alias:	buzzreleasekey
Password:	•••••••
Confirm:	•••••••
Validity (years):	25
First and Last Name:	Buzztouch User
Organizational Unit:	Development
Organization:	CompanyName
City or Locality:	Sometown
State or Province:	AZ
Country Code (XX):	US

g. Now you'll want to select a destination for your .apk file. As always, make sure it's somewhere you remember. You can also give your package a name at this step.


Export Android Application

Destination and key/certificate checks

Destination APK file: C:\Documents and Settings\mf\Desktop\buzzTouchRele

Certificate expires in 25 years.

h. Hit **Finish**, and the release package for your application will be built! You'll also now have a release keystore good for 25 years, and you can use it to request your Google Maps API key!

So, this has been a bit of a long process to get ready to request our MD5 fingerprints, but we're ready for that now. This is a command line procedure, and is accomplished as follows:

1. **Get the MD5 Fingerprint of your Signing Certificate.** From the command line or terminal session, enter the following command.

```
a. keytool -list -alias your_release_key_alias -keystore
   your_release.keystore -storepass your_password -keypass
   your_password
```

The result of this command should look something like this:

Certificate fingerprint (MD5): 91:EF:AA:33:DD:CC:C2:1D:7F:AB:9C:49:DC:36:D9:3B

2. **Get the MD5 Fingerprint of your SDK Debug Certificate.** From the command line or terminal session, enter the following command.

```
a. keytool -list -alias androiddebugkey -keystore
   debug.keystore -storepass android -keypass android
```

The result of this command should look something like this:

Certificate fingerprint (MD5): G1:88:33:77:99:22:7F:A1:A1:47:87:10:90:C6:53:6C

At this point, you have all the information you need to request your Google Maps API debug and release keys! Follow these steps, and you'll be up and running soon!

1. Go to http://code.google.com/android/maps-api-signup.html
2. If you don't have a Google account, use the link on the page to set one up.
3. Read the Android Maps API Terms of Service carefully. If you agree to the terms, indicate so using the checkbox on the screen.
4. Paste the MD5 certificate fingerprint of the certificate that you are registering into the appropriate form field.
5. Click "Generate API Key"

The website will generate a code that looks something like this:

0oYoJTnGQYNBciyXYZfbutxABCBqAsEjshrbpIw

Get an API key for **both** your debug and release MD5 fingerprints and write them down for safe keeping. I created a file that contains my MD5 fingerprints (along with the command I used to get them) and the corresponding API key. That way I have all the info I need in one place.

You're now ready for the last step, which is configuring your Android application with the correct keys. Double check the Readme.txt file that comes with your Android project package from buzztouch, but at the time of this writing, the location for placing your keys is in the res\values\string.xml file. If you open it up, you'll see the following two lines near the top:

```
<string name="googleMapsAPIKeyDebug">yourGoogleMapsKeyForDebug</string>

<string name="googleMapsAPIKeyRelease">yourGoogleMapsKeyForRelease</string>
```

Paste your keys in the appropriate line, and you’re good to go!

Chapter 11

Publishing Your App to Google Play

The moment you've been waiting for has finally arrived. You've designed your app, tested it, tweaked it, and created the final file. There's only one more step left – time to publish!

There are multiple ways you could distribute your Android app:

- Publish on Google Play
- Publish on Amazon Marketplace
- Email your app to people
- Post it on a website
- A jillion other methods!

This chapter will focus on publishing your app to Google Play, which is arguably the most popular way to get your app onto the millions (literally!) of Android devices across the world.

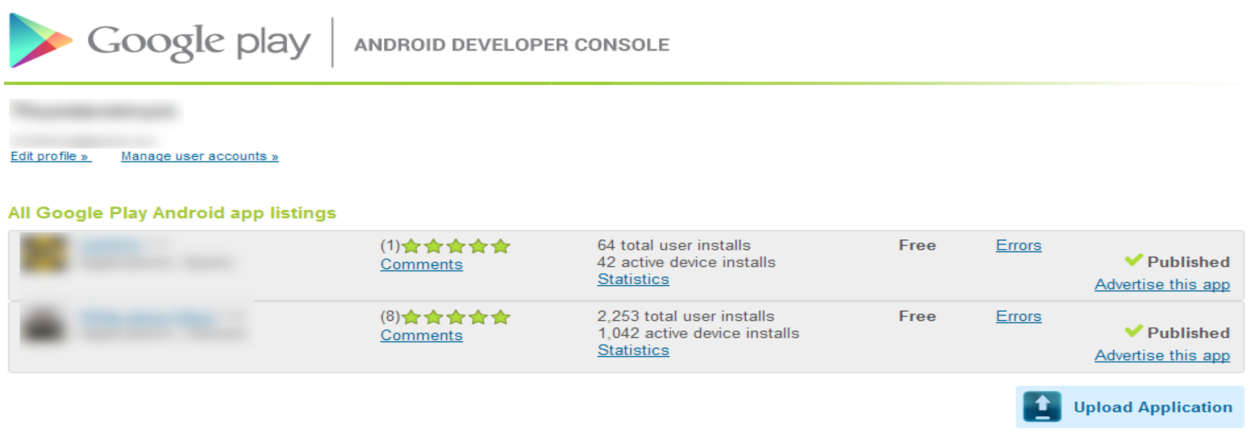
Ready? Let's get started!

First thing you need to do is head over to the following URL and set up your **Google Play Android Developer Account**. Remember…it'll cost you $25 to sign up (onetime fee).

https://play.google.com/apps/publish

If you already have a Google account (any of them), this will be super easy.

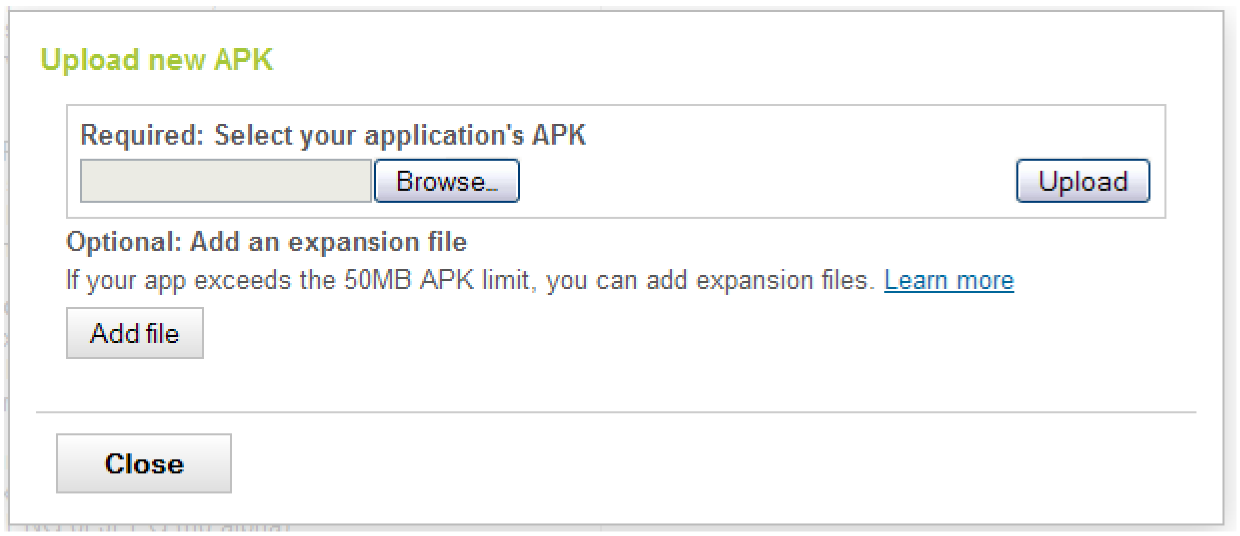
Once you log into your Developer Console, you'll be presented with a screen that looks similar to this. It'll be different if you already have apps published.



All Google Play Android app listings

[illegible]	(1) ★★★★★ Comments	64 total user installs 42 active device installs Statistics	Free	Errors	✓ Published Advertise this app
[illegible]	(8) ★★★★★ Comments	2,253 total user installs 1,042 active device installs Statistics	Free	Errors	✓ Published Advertise this app

Click the **Upload Application** button to start the process of submitting a new app. You'll see this screen:



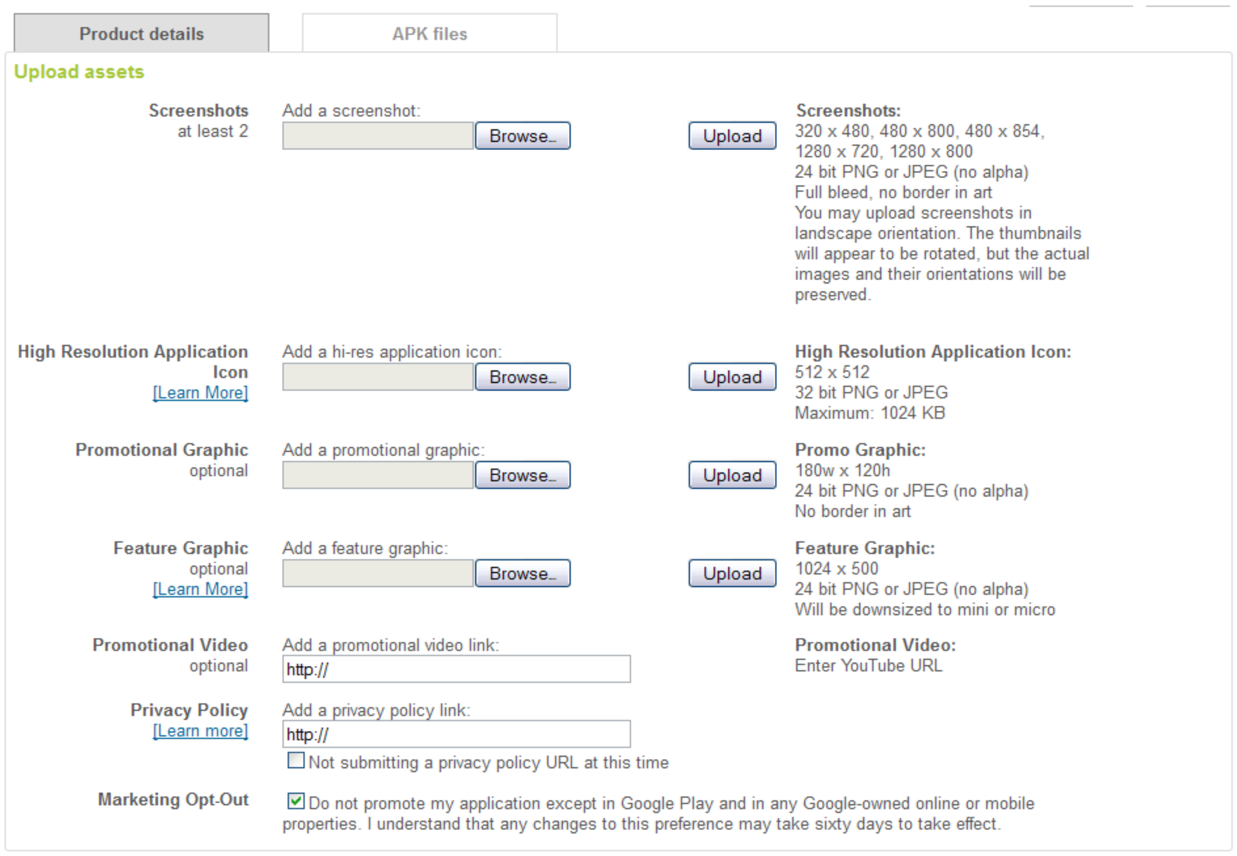
Upload new APK

Required: Select your application's APK

Optional: Add an expansion file

If your app exceeds the 50MB APK limit, you can add expansion files. Learn more

If for some reason you are not ready with your .apk file, that's OK…just hit **Close**. It'll then take you to the next screen in the process. If you do have your .apk file ready, hit **Browse**, navigate to the file, and then hit **Upload**. When it's done, you'll get directed to the same screen that you would if you had hit Close.



Upload assets

Screenshots at least 2	Add a screenshot:	Screenshots: 320 x 480, 480 x 800, 480 x 854, 1280 x 720, 1280 x 800 24 bit PNG or JPEG (no alpha) Full bleed, no border in art You may upload screenshots in landscape orientation. The thumbnails will appear to be rotated, but the actual images and their orientations will be preserved.
High Resolution Application Icon [Learn More]	Add a hi-res application icon:	High Resolution Application Icon: 512 x 512 32 bit PNG or JPEG Maximum: 1024 KB
Promotional Graphic optional	Add a promotional graphic:	Promo Graphic: 180w x 120h 24 bit PNG or JPEG (no alpha) No border in art
Feature Graphic optional [Learn More]	Add a feature graphic:	Feature Graphic: 1024 x 500 24 bit PNG or JPEG (no alpha) Will be downsized to mini or micro
Promotional Video optional	Add a promotional video link: http://	Promotional Video: Enter YouTube URL
Privacy Policy [Learn more]	Add a privacy policy link: http:// ☐ Not submitting a privacy policy URL at this time	
Marketing Opt-Out	☑ Do not promote my application except in Google Play and in any Google-owned online or mobile properties. I understand that any changes to this preference may take sixty days to take effect.	

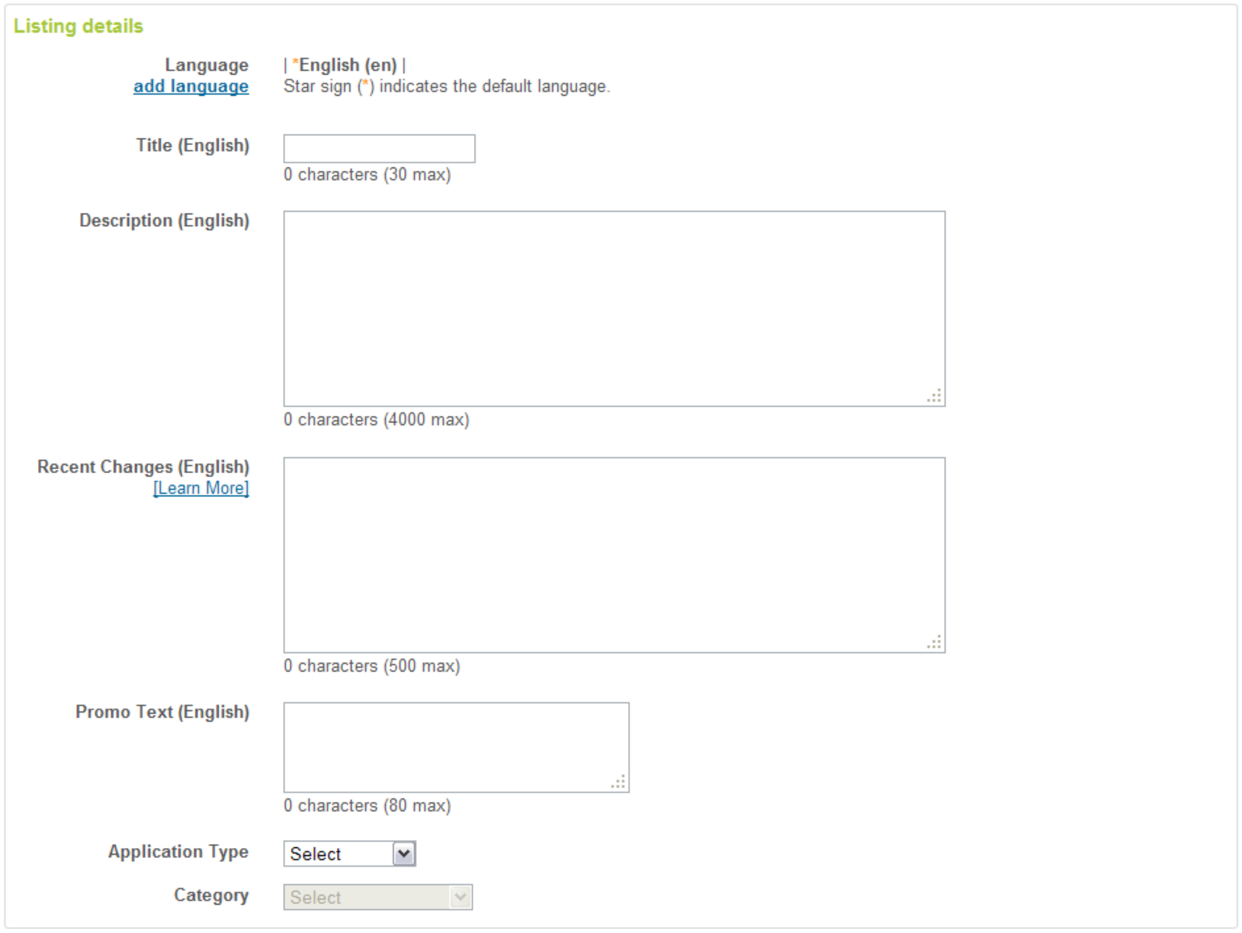
This is the first part of a longer screen, but it's really the meat of what you need to provide in order to publish your app. Here's a synopsis of what you'll need to provide (these are required):

1. At least two (2) screenshots. They need to be in either .png or .jpg format, and should be in one of the following dimensions:
    a. 320x480,
    b. 480x800
    c. 480x854
    d. 1280x720
    e. 1280x800
2. A high resolution application icon with 512x512 dimensions
3. A title for your app (30 characters max)
4. A description in English of your app (what do you want your users to know about it?)
5. An Application Type
6. A Category
7. Copy Protection decision
    a. Off (allows the app to be copied from the device)
    b. On (helps prevent copying)
8. A content rating
9. Price (Free, unless you configure a Merchant Account)
10. Countries you want it available in
11. Contact information (be sure to provide at least one)
    a. Website
    b. Email
    c. Phone
12. Acknowledgement of a couple consent items

Once you provide all that information, upload the .apk if you haven't already, you then have two choices:

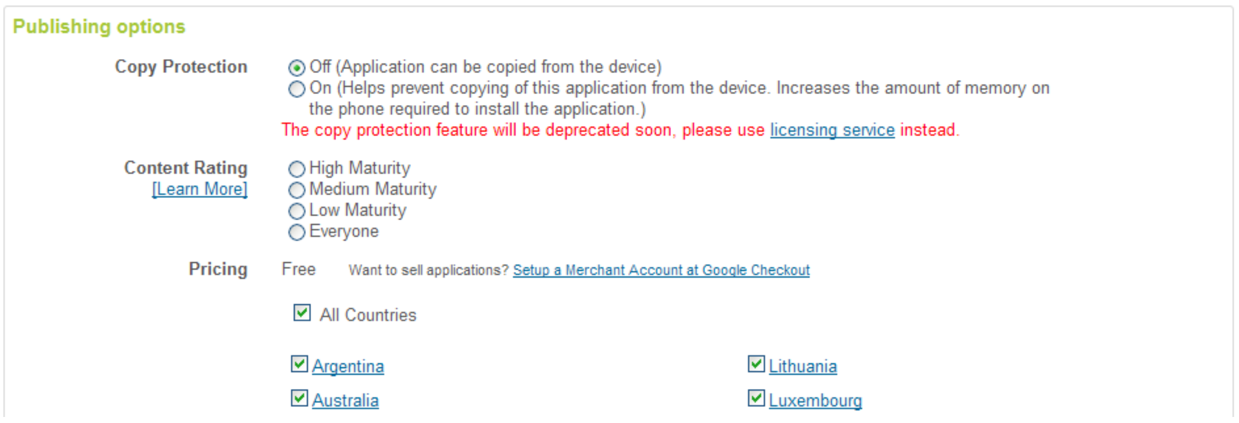
1. **Publish** – Make it available to the world!
2. **Save** – Save the info until you have everything in order.

Here are some screen shots of what we were talking about in the steps on the previous page:



Listing details

Language add language	| *English (en) | Star sign (*) indicates the default language.
Title (English)	0 characters (30 max)
Description (English)	0 characters (4000 max)
Recent Changes (English) [Learn More]	0 characters (500 max)
Promo Text (English)	0 characters (80 max)
Application Type	Select
Category	Select


Publishing options

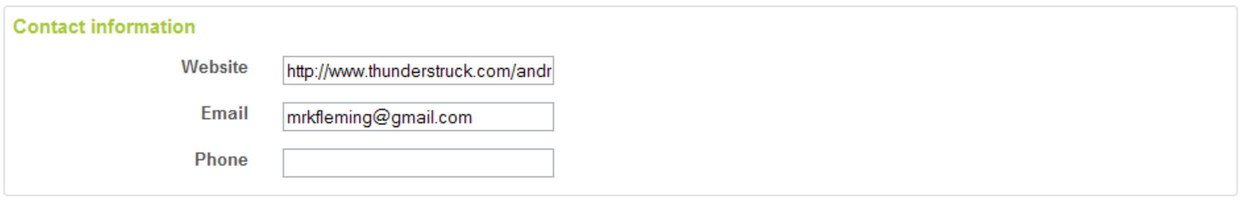
Copy Protection [✓ Off (Application can be copied from the device)] On (Helps prevent copying of this application from the device. Increases the amount of memory on the phone required to install the application.)
The copy protection feature will be deprecated soon, please use licensing service instead.

Content Rating [Learn More]: High Maturity / Medium Maturity / Low Maturity / Everyone

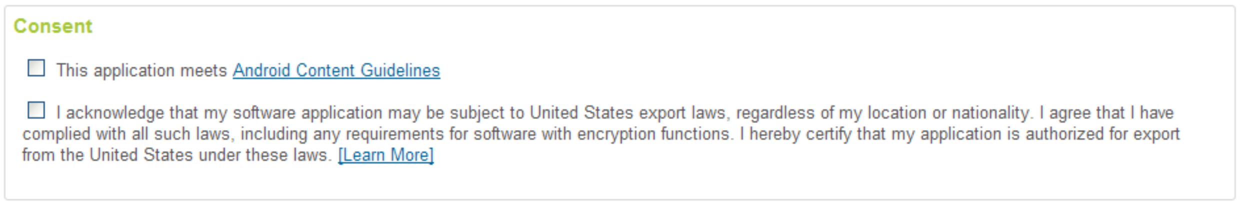
Pricing: Free — Want to sell applications? Setup a Merchant Account at Google Checkout

✓ All Countries

✓ Argentina	✓ Lithuania
✓ Australia	✓ Luxembourg


Contact information

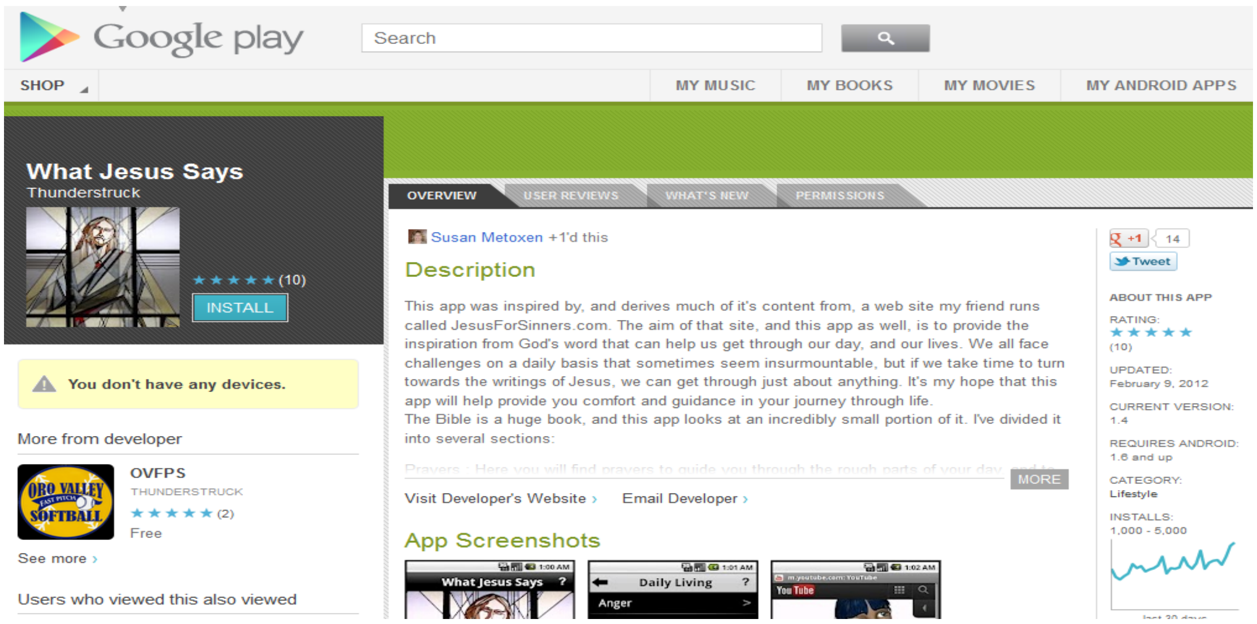
Website	http://www.thunderstruck.com/andr
Email	mrkfleming@gmail.com
Phone	


Consent

☐ This application meets Android Content Guidelines

☐ I acknowledge that my software application may be subject to United States export laws, regardless of my location or nationality. I agree that I have complied with all such laws, including any requirements for software with encryption functions. I hereby certify that my application is authorized for export from the United States under these laws. [Learn More]

Bam! That's it! In an amount of time that seems to range from a couple minutes to a couple hours, your app will be live at Google Play, and available for download by literally millions of people! Here's a screen shot of what it might look like. Congratulations on being a published app developer!



What Jesus Says
Thunderstruck
★★★★★ (10)

You don't have any devices.

More from developer
OVFPS
THUNDERSTRUCK
★★★★★ (2)
Free
See more ›

Users who viewed this also viewed

Susan Metoxen +1'd this

Description

This app was inspired by, and derives much of it's content from, a web site my friend runs called JesusForSinners.com. The aim of that site, and this app as well, is to provide the inspiration from God's word that can help us get through our day, and our lives. We all face challenges on a daily basis that sometimes seem insurmountable, but if we take time to turn towards the writings of Jesus, we can get through just about anything. It's my hope that this app will help provide you comfort and guidance in your journey through life.
The Bible is a huge book, and this app looks at an incredibly small portion of it. I've divided it into several sections:

Visit Developer's Website › Email Developer ›

App Screenshots

What Jesus Says
Daily Living
Anger

ABOUT THIS APP
RATING: ★★★★★ (10)
UPDATED: February 9, 2012
CURRENT VERSION: 1.4
REQUIRES ANDROID: 1.6 and up
CATEGORY: Lifestyle
INSTALLS: 1,000 - 5,000

Chapter 12

Common Problems and Solutions

What follows below is a random listing of some common Android-related problems you may face, and their solutions. This list will grow as we produce more apps and run into more issues!

- **Problem:** AndroidManifest.xml is missing.
    1. The reason for this error is currently unknown. Here are some potential solutions.
        - Turn off everything and restart
        - Verify your installation of Eclipse
        - Verify AndroidManifest.xml actually exists in Package Explorer
        - Right-click on Project, select Android Tools…Build Project Properties
- **Problem:** Project has no default.properties file! Edit the project properties to set one.
    1. There are multiple reasons for this, but it basically refers to a problem with your default.properties file. The easiest way to fix this is to **right-click** on your project in Project Explorer, and pick **Android Tools…Fix Project Properties**.
- **Problem:** “Android requires compiler compliance level 5.0 or 6.0. Found '1.7' instead. Please use Android Tools > Fix Project Properties.”
    1. For whatever reason, your project properties are not set correctly.
        - Right-click on your project and select "Properties -> Java Compiler", check "Enable project specific settings" and select 1.5 or 1.6 from "Compiler compliance settings" select box. Try different levels if the first one does not work.
    2. http://stackoverflow.com/questions/7637144/android-requires-compiler-compliance-level-5-0-or-6-0-found-1-7-instead-plea
- **Problem:** “Installation error: INSTALL_FAILED_INSUFFICIENT_STORAGE”
    1. It seems like there are a gagillion reasons for this, and no real solutions, but apparently increasing the amount of memory available to your AVD works in many instances.
        - *Solution #1 :* Find the `.android` folder on your system. That’s where your AVD configuration files are. In my Windows system, it’s located at `C:\Books and Settings\mf\.android\avd\.` Open up the <avdname>.avd file and

put this line in it : `disk.dataPartition.size=2048`. What this apparently does it change the size of the partition dedicated to the AVD to 2Gb, which appears to be more than enough for most apps. Since editing this file, I have not received the INSTALL_FAILED_INSUFFICIENT_STORAGE message.

    - *Solution #2:* Right click the root of your Android Project, go to "Run As" then go to "Run Configurations..." locate the "Android Application" node in the tree at the left, then select your project and go to the "Target" tab on the right side of the window look down for the "Additional Emulator Command Line Options" field (sometimes you'll need to make the window larger) and finally paste "-partition-size 1024" there. Click Apply and then Run to use your emulator.

    - *Solution #3 :* Go to Eclipse's Preferences, then Select “Launch” Add “-partition-size 1024” on the “Default emulator option” field, then click “Apply” and use your emulator as usual.

  2. http://stackoverflow.com/questions/4709137/solution-android-install-failed-insufficient-storage-error

  3. http://stackoverflow.com/questions/2239330/how-to-increase-storage-for-android-emulator-install-failed-insufficient-stora

- **Problem**: Your Android v1.5 or v2.0 app has a map screen in it, and the Google Map tiles are not showing up when the screen is launched. All the pins are there, but no map tiles. Often times this works in the emulator, but then fails to work when you publish the app. Using a combination of experience and some info from @Fred’s blog (http://myskylla.wordpress.com/), here are the steps to configure maps (not including driving directions) for your Android app.

  1. Obtain a Google Maps API Key. You can use this book to get you through that process: http://www.buzztouch.com/resources/Obtaining_a_Google_Maps_API_Key_v1.0.pdf

  2. Copy and paste the Release and Debug API Keys into your `\res\values\strings.xml` file. The lines you need are near the top of the file, and look like the following. Once done, save and close the file.

     - `<string name="googleMapsAPIKeyDebug">DebugKey</string>`

     - `<string name="googleMapsAPIKeyRelease">ReleaseKey</string>`

  3. In the \res\layout\ screen_map.xml file, change the line that reads `android:apiKey="@string/googleMapsAPIKeyDebug"` **TO** `android:apiKey="@string/googleMapsAPIKey`**`Release`**`"` (notice all that has changed is the word Debug to Release)

4. In your AndroidManifest.xml file, which is in the root folder of your project, uncomment the lines near the bottom of the file that look like the following. For clarity, check the README file that came with your project.

   ```
   <!-- required libraries for maps -->

   <uses-library android:name="com.google.android.maps"/>
   ```

5. Save everything, compile your app, sign it using your Release key, and publish!

Chapter 13

Problem Reporting and Basic Debug

We've all seen the post in the forum that goes something like this:

"Tried to compile my app, and it doesn't work! Please help!"

Or maybe it goes something like this:

"Got errors when I tried to compile my app, and it won't run on my device. What do I do?"

The people who participate in the buzztouch forum are some of the most amazing and intelligent people I have ever been involved with, but the only way they can help you fix your problems is if you give as much information as possible in your problems reports, and you provide as much debug help as possible. Without some basic information in the initial post, we all have to spend time asking a set of questions that should **always** be answered right from the start. I believe that the following should be included in every problem report you put in the forums:

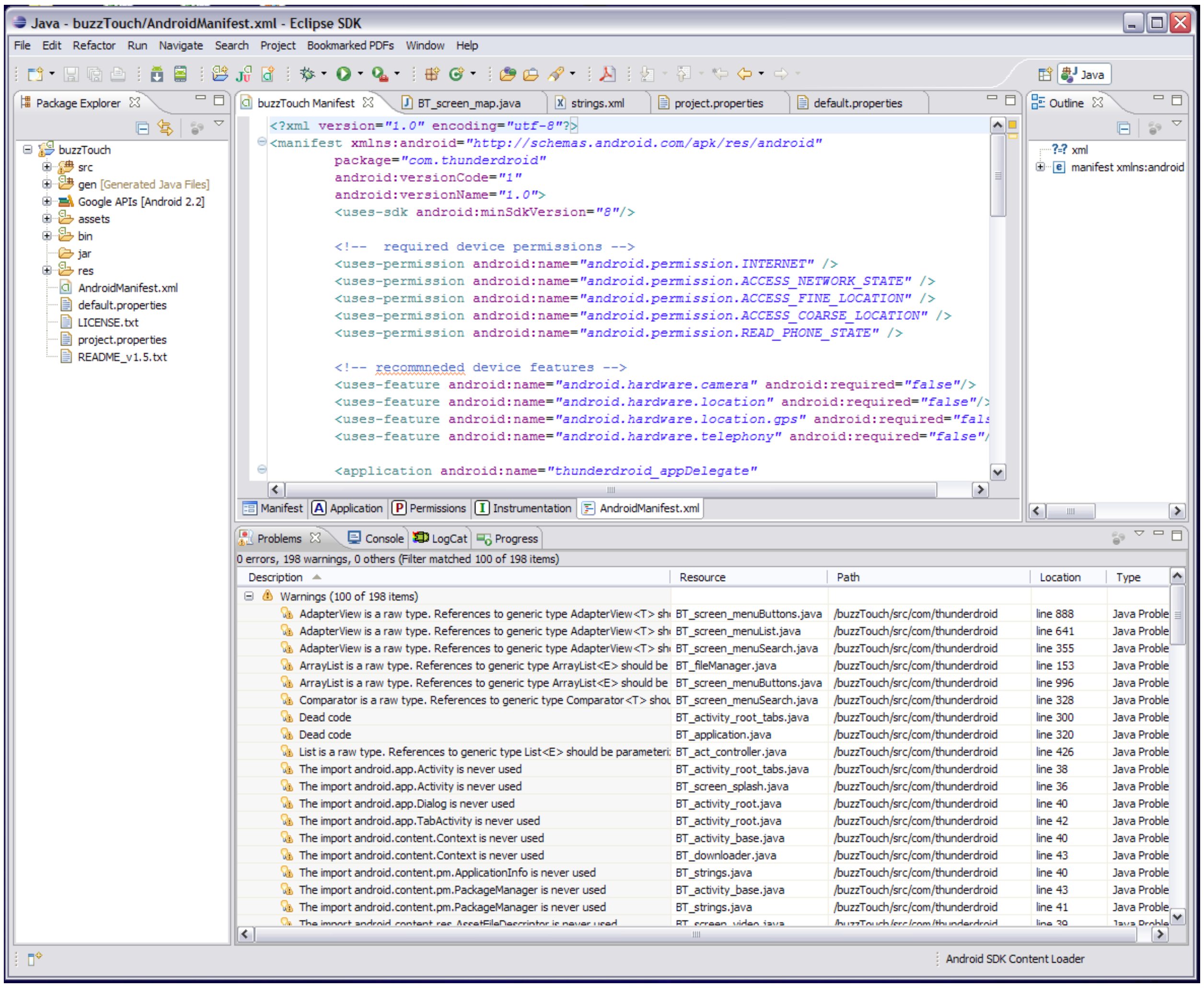
1. buzztouch Version : **v1.4** or **v1.5**
2. **Android**, **iOS** or **Both**
3. What version of **Xcode/iOS SDK?**
4. What version of **Eclipse/Android SDK?**
5. The **most complete problem description** you can give. More details are better than fewer details, and actual error messages pasted into the post are very helpful. Don't just say it doesn't work or that there are errors…tell us **how** it's not working, and **what** the actual error messages are!
6. **Screen captures** so we can see what's going on are often extremely helpful. Remember the saying; a picture is worth a thousand words! DropBox is a great place to host images, and best of all it's FREE! (use this link to sign up and I'll get an extra 250MB of space free when you do! http://db.tt/TCxVvk8)

You will be amazed and how much help you will get from your fellow forum users if you provide this basic information. Lots of times we can look at what you post and the answer is obvious because we've run into the same issues before ourselves. But sometimes there's literally not enough information to even begin to debug the problem, and it's highly unlikely that a person will visit your post again to see if you've updated it with more info. Do it right the first time, and you'll get your problem solved quicker!

Now that you know what sort of information is most helpful to people in the forum, let's take a look at how to set up your environment to show helpful debug info. Both Xcode and Eclipse provide some tools that let you know exactly what's going on with every step of your app compiling and running. I'll show you how to configure some of that now.

Configuring Eclipse For Debug Information

Eclipse provides many useful options for debugging your application. Here's what a typical screen might look like running Eclipse 3.7.1 (Indigo) that has the Android SDK installed in it.



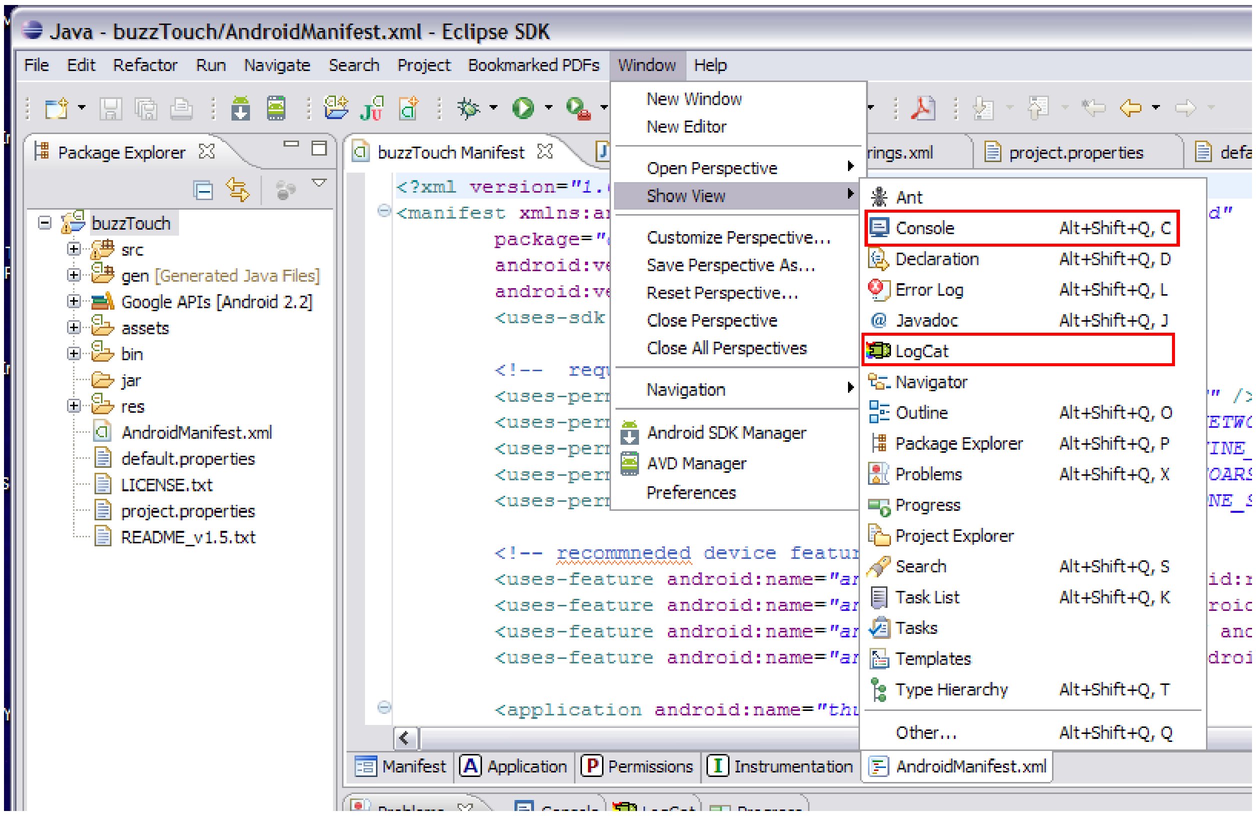
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.thunderdroid"
        android:versionCode="1"
        android:versionName="1.0">
        <uses-sdk android:minSdkVersion="8"/>

        <!--  required device permissions -->
        <uses-permission android:name="android.permission.INTERNET" />
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />

        <!-- recommneded device features -->
        <uses-feature android:name="android.hardware.camera" android:required="false"/>
        <uses-feature android:name="android.hardware.location" android:required="false"/>
        <uses-feature android:name="android.hardware.location.gps" android:required="fals
        <uses-feature android:name="android.hardware.telephony" android:required="false"/

        <application android:name="thunderdroid_appDelegate"
```

0 errors, 198 warnings, 0 others (Filter matched 100 of 198 items)

Description	Resource	Path	Location	Type
Warnings (100 of 198 items)				
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuButtons.java	/buzzTouch/src/com/thunderdroid	line 888	Java Proble
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuList.java	/buzzTouch/src/com/thunderdroid	line 641	Java Proble
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuSearch.java	/buzzTouch/src/com/thunderdroid	line 355	Java Proble
ArrayList is a raw type. References to generic type ArrayList<E> should be	BT_fileManager.java	/buzzTouch/src/com/thunderdroid	line 153	Java Proble
ArrayList is a raw type. References to generic type ArrayList<E> should be	BT_screen_menuButtons.java	/buzzTouch/src/com/thunderdroid	line 996	Java Proble
Comparator is a raw type. References to generic type Comparator<T> shou	BT_screen_menuSearch.java	/buzzTouch/src/com/thunderdroid	line 328	Java Proble
Dead code	BT_activity_root_tabs.java	/buzzTouch/src/com/thunderdroid	line 300	Java Proble
Dead code	BT_application.java	/buzzTouch/src/com/thunderdroid	line 320	Java Proble
List is a raw type. References to generic type List<E> should be parameteri:	BT_act_controller.java	/buzzTouch/src/com/thunderdroid	line 426	Java Proble
The import android.app.Activity is never used	BT_activity_root_tabs.java	/buzzTouch/src/com/thunderdroid	line 38	Java Proble
The import android.app.Activity is never used	BT_screen_splash.java	/buzzTouch/src/com/thunderdroid	line 36	Java Proble
The import android.app.Dialog is never used	BT_activity_root.java	/buzzTouch/src/com/thunderdroid	line 40	Java Proble
The import android.app.TabActivity is never used	BT_activity_root.java	/buzzTouch/src/com/thunderdroid	line 42	Java Proble
The import android.content.Context is never used	BT_activity_base.java	/buzzTouch/src/com/thunderdroid	line 40	Java Proble
The import android.content.Context is never used	BT_downloader.java	/buzzTouch/src/com/thunderdroid	line 43	Java Proble
The import android.content.pm.ApplicationInfo is never used	BT_strings.java	/buzzTouch/src/com/thunderdroid	line 40	Java Proble
The import android.content.pm.PackageManager is never used	BT_activity_base.java	/buzzTouch/src/com/thunderdroid	line 43	Java Proble
The import android.content.pm.PackageManager is never used	BT_strings.java	/buzzTouch/src/com/thunderdroid	line 41	Java Proble
The import android.content.res.AssetFileDescriptor is never used	BT_screen_video.java	/buzzTouch/src/com/thunderdroid	line 39	Java Proble

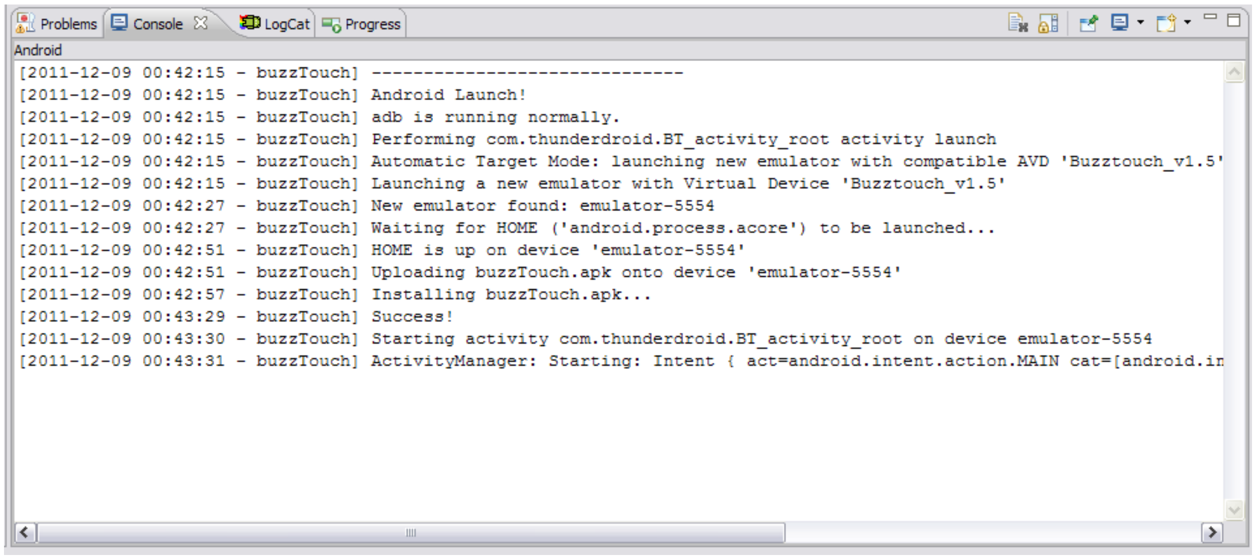
There's a lot going on here, and you can add or subtract what you want by clicking on the X for a particular pane, or by adding views and such through the tool bar. Just click around and you'll eventually get things configured the way you want them.

The area at the bottom of your screen is where you'll find most of the debug screens that will be helpful to you as you try and figure out what's going on in your application. If for some reason you don't see the same panes that I do, you can enable them pretty easily by going to **Window…Show View** and selecting the views that you wish to see. Here's what that looks like:



I've highlighted two views…**Console** and **LogCat**…as being the most useful for debugging. Let's take a look at what each does.

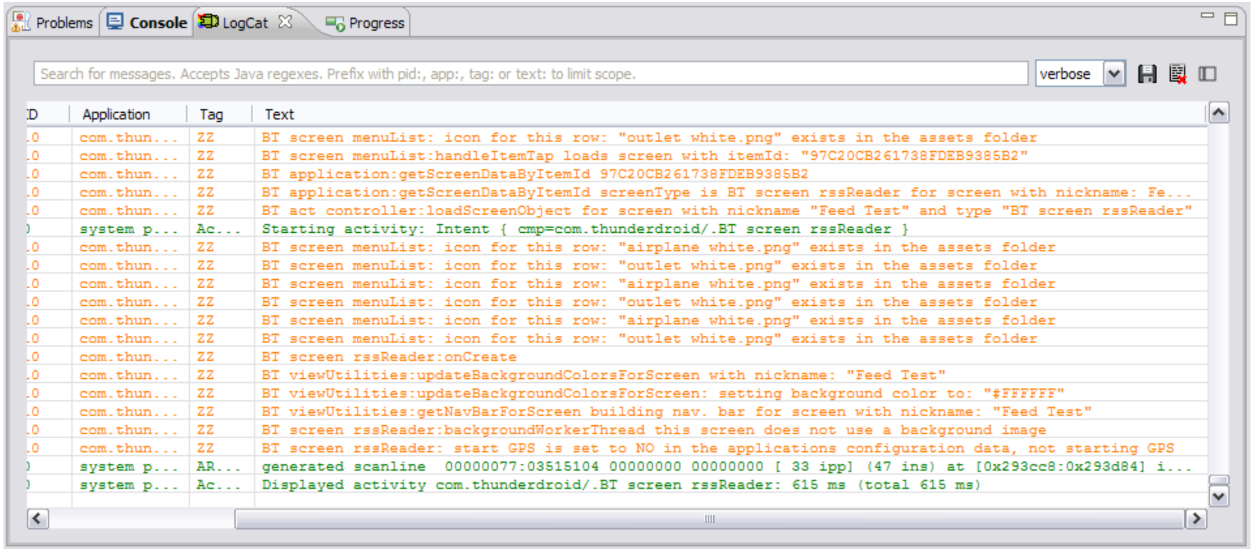
The Console view pretty much gives you an indication on whether your app is being loaded into the Android Virtual Device (AVD) you created and tied to your app. If there are problems launching the console, installing the app, or anything related to that, those will be logged here. Here's an example of what you hope to see in the Console, because, in this case, it means your app loaded successfully and you're off and running!



```
Android
[2011-12-09 00:42:15 - buzzTouch] ------------------------------
[2011-12-09 00:42:15 - buzzTouch] Android Launch!
[2011-12-09 00:42:15 - buzzTouch] adb is running normally.
[2011-12-09 00:42:15 - buzzTouch] Performing com.thunderdroid.BT_activity_root activity launch
[2011-12-09 00:42:15 - buzzTouch] Automatic Target Mode: launching new emulator with compatible AVD 'Buzztouch_v1.5'
[2011-12-09 00:42:15 - buzzTouch] Launching a new emulator with Virtual Device 'Buzztouch_v1.5'
[2011-12-09 00:42:27 - buzzTouch] New emulator found: emulator-5554
[2011-12-09 00:42:27 - buzzTouch] Waiting for HOME ('android.process.acore') to be launched...
[2011-12-09 00:42:51 - buzzTouch] HOME is up on device 'emulator-5554'
[2011-12-09 00:42:51 - buzzTouch] Uploading buzzTouch.apk onto device 'emulator-5554'
[2011-12-09 00:42:57 - buzzTouch] Installing buzzTouch.apk...
[2011-12-09 00:43:29 - buzzTouch] Success!
[2011-12-09 00:43:30 - buzzTouch] Starting activity com.thunderdroid.BT_activity_root on device emulator-5554
[2011-12-09 00:43:31 - buzzTouch] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.in
```

If for some reason the AVD has a problem initializing and launching, or the app won't install to the AVD, you'll get errors logged here.

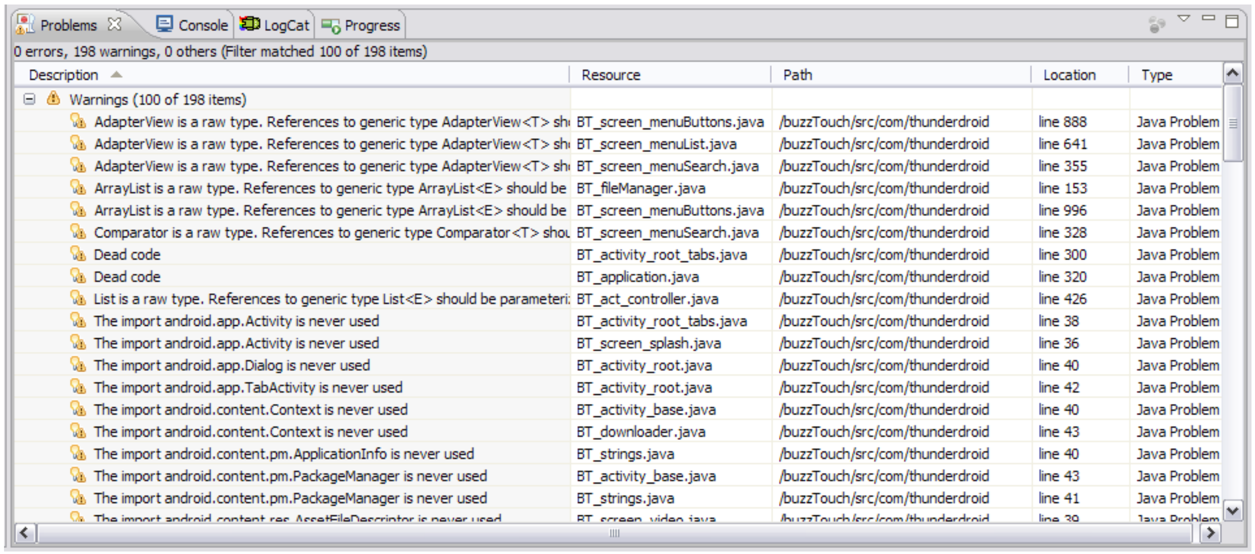
The next place of interest is **LogCat**. This view logs pretty much every move made in your app. It reports on the existence, or non-existence of files or other resources. It tells you which files are being loaded, and from where. It tells you if there was an error, and what it was. If you experience a problem with your app, this is the first place you should look for information on what went wrong. Here's a peek at what you can expect to find.



```
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.   verbose

D    Application   Tag    Text
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "outlet white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList:handleItemTap loads screen with itemId: "97C20CB261738FDEB9385B2"
.0   com.thun...   ZZ     BT application:getScreenDataByItemId 97C20CB261738FDEB9385B2
.0   com.thun...   ZZ     BT application:getScreenDataByItemId screenType is BT screen rssReader for screen with nickname: Fe...
.0   com.thun...   ZZ     BT act controller:loadScreenObject for screen with nickname "Feed Test" and type "BT screen rssReader"
)    system p...   Ac...  Starting activity: Intent { cmp=com.thunderdroid/.BT screen rssReader }
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "airplane white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "outlet white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "airplane white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "outlet white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "airplane white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen menuList: icon for this row: "outlet white.png" exists in the assets folder
.0   com.thun...   ZZ     BT screen rssReader:onCreate
.0   com.thun...   ZZ     BT viewUtilities:updateBackgroundColorsForScreen with nickname: "Feed Test"
.0   com.thun...   ZZ     BT viewUtilities:updateBackgroundColorsForScreen: setting background color to: "#FFFFFF"
.0   com.thun...   ZZ     BT viewUtilities:getNavBarForScreen building nav. bar for screen with nickname: "Feed Test"
.0   com.thun...   ZZ     BT screen rssReader:backgroundWorkerThread this screen does not use a background image
.0   com.thun...   ZZ     BT screen rssReader: start GPS is set to NO in the applications configuration data, not starting GPS
)    system p...   AR...  generated scanline  00000077:03515104 00000000 00000000 [ 33 ipp] (47 ins) at [0x293cc8:0x293d84] i...
)    system p...   Ac...  Displayed activity com.thunderdroid/.BT screen rssReader: 615 ms (total 615 ms)
```

As I indicated, there is tons and tons of seriously useful information here, and it can all be copied and pasted to the forum for review by others. I don't think I would copy everything…that would be way too much! The best approach here is to do something in the AVD that isn't working, and copy the information that gets logged as a result of that action. Then you can easily correlate your actions with particular failures. It's absolutely worth your time to play around with LogCat and see how amazing it is…you'll probably learn a lot about your app that you never knew!

The final section I want to mention is the **Problems** view. This view gives you an indication of any Warnings or Problems that might prevent your app from compiling or running correctly. It looks pretty much like this:



0 errors, 198 warnings, 0 others (Filter matched 100 of 198 items)

Description	Resource	Path	Location	Type
Warnings (100 of 198 items)				
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuButtons.java	/buzzTouch/src/com/thunderdroid	line 888	Java Problem
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuList.java	/buzzTouch/src/com/thunderdroid	line 641	Java Problem
AdapterView is a raw type. References to generic type AdapterView<T> sh	BT_screen_menuSearch.java	/buzzTouch/src/com/thunderdroid	line 355	Java Problem
ArrayList is a raw type. References to generic type ArrayList<E> should be	BT_fileManager.java	/buzzTouch/src/com/thunderdroid	line 153	Java Problem
ArrayList is a raw type. References to generic type ArrayList<E> should be	BT_screen_menuButtons.java	/buzzTouch/src/com/thunderdroid	line 996	Java Problem
Comparator is a raw type. References to generic type Comparator<T> shou	BT_screen_menuSearch.java	/buzzTouch/src/com/thunderdroid	line 328	Java Problem
Dead code	BT_activity_root_tabs.java	/buzzTouch/src/com/thunderdroid	line 300	Java Problem
Dead code	BT_application.java	/buzzTouch/src/com/thunderdroid	line 320	Java Problem
List is a raw type. References to generic type List<E> should be parameteri	BT_act_controller.java	/buzzTouch/src/com/thunderdroid	line 426	Java Problem
The import android.app.Activity is never used	BT_activity_root_tabs.java	/buzzTouch/src/com/thunderdroid	line 38	Java Problem
The import android.app.Activity is never used	BT_screen_splash.java	/buzzTouch/src/com/thunderdroid	line 36	Java Problem
The import android.app.Dialog is never used	BT_activity_root.java	/buzzTouch/src/com/thunderdroid	line 40	Java Problem
The import android.app.TabActivity is never used	BT_activity_root.java	/buzzTouch/src/com/thunderdroid	line 42	Java Problem
The import android.content.Context is never used	BT_activity_base.java	/buzzTouch/src/com/thunderdroid	line 40	Java Problem
The import android.content.Context is never used	BT_downloader.java	/buzzTouch/src/com/thunderdroid	line 43	Java Problem
The import android.content.pm.ApplicationInfo is never used	BT_strings.java	/buzzTouch/src/com/thunderdroid	line 40	Java Problem
The import android.content.pm.PackageManager is never used	BT_activity_base.java	/buzzTouch/src/com/thunderdroid	line 43	Java Problem
The import android.content.pm.PackageManager is never used	BT_strings.java	/buzzTouch/src/com/thunderdroid	line 41	Java Problem
The import android.content.res.AssetFileDescriptor is never used	BT_screen_video.java	/buzzTouch/src/com/thunderdroid	line 39	Java Problem

In this particular example, all we see here are Warnings, and not all Warnings are things you need to be concerned about, and they generally won't prevent your app from compiling. However, if you see some Problems listed here, they **will** prevent your app from being compiled, and they need to be resolved before you can move on. If you experience weird app behavior, it's probably worth a few minutes to look through this view.

That pretty much sums up the basics for Eclipse! Between those three views, you should have plenty of great debug information to help you solve your problem, or to post to the forum so others can help.

Chapter 14

Final Thoughts

With your discovery of buzztouch.com, and your dedication to making it through this way too long book, you have started on a journey you probably never imagined, and one you'll probably never stop being on. There's nothing quite like spending hours and hours in front of your computer, piecing together the parts of what you know is going to be an amazing app, then publishing it and watching the download count increase! It's an accomplishment you should be very, very proud of.

Good luck with your app making. If you have some time, consider dropping a line to David, Parker and Stephen at buzztouch (info@buzztouch.com) letting them know what an awesome product they have. I'm sure they'd appreciate it.

Comments? Post them in the forum or email me at MrkFleming@gmail.com.

A list of current book versions can be found here:
https://dl.dropbox.com/u/4993711/CurrentBTDocVersions.txt

Revision Log			
v1.0	06/16/12	Initial release of book.	GoNorthWest