Google Android Development

Lesson #1
Introduction

Welcome!

Over the next seven sessions, you're going to be introduced to the concepts, techniques, and technologies necessary to create applications for the Android platform.

We'll look at:

- The development tools, SDKs, and installations of required components.
- Basics of programming in Java.
- Basics of the Eclipse IDE (Integrated Development Environment)
- Android UI (User Interface) elements, and how they work
- Android Events, Programming Concepts, and Android's capabilities.
- We'll build sample apps throughout the process, and go through the steps necessary to publish an application in the Android Marketplace.

Equipment?!

In order to write applications for Android, you do not need to have an Android device. Development can be done without spending a dime on software or additional hardware. All you really need is a computer running Windows, Apple OS X, or Linux.

If you happen to already own an Android phone, such as a Motorala Droid, Google Nexus One, or HTC Evo 4G, you can install the applications you write directly onto that device, and test them.

Programming

Developing applications for a mobile platform is very different from traditional desktop, server, or web application development. Mobile applications tend to be much more focused on a specific task, and typically do not provide the broad set of features that a desktop application would.

Writing applications for Android is easier than for other mobile operating systems, such as Apple's iOS, or Nokia's Symbian. Of course any programming scenario brings its own challenges and hurdles to overcome.

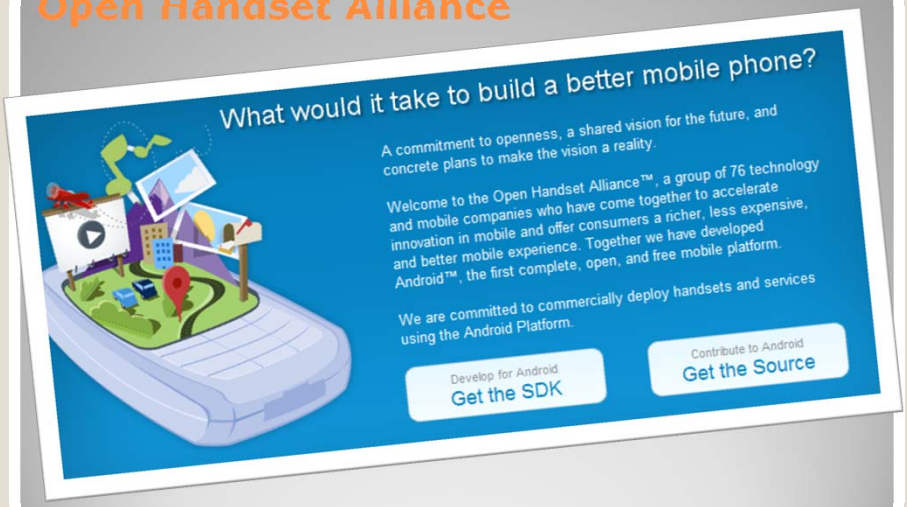
If you're new to programming, writing Android applications is a great way to start!

Hello Android!

Android is an operating system based on Linux, developed by Google and the Open Handset Alliance. It is designed to run on embedded and mobile devices of all types. While today we see Android primarily on cellular telephones, it is likely that we'll see it integrated into a variety of electronic devices, such as televisions, cars, e-book readers, and more.

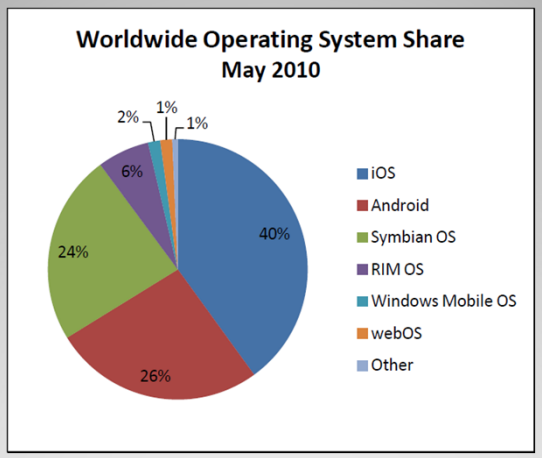
Because Android is free, and open source, it can be used freely by just about any hardware manufacturer. By utilizing Android, they save on the cost of developing their own proprietary OS for their device, and can take advantage of numerous Android benefits, such as the Android Marketplace, OS updates, and various features within the OS.

Open Handset Alliance



The Open Handset Alliance is led by Google, Inc.

Android Market Share - 2010



iOS (Apple iPhone OS) has been around for 4 years, while Android was introduced in November of 2008.

http://metrics.admob.com/wp-content/uploads/2010/06/May-2010-AdMob-Mobile-Metrics-Highlights.pdf

Requirements

PC Requirements:

- Windows XP, Vista, or Windows 7 (32bit or 64bit)
- Mac OS X 10.5.8 or higher (Intel platform only)
- Linux (32bit or 64bit)

Note: Even though 64bit operating systems are supported, it is preferable to use a 32bit versions of all of the software components necessary.

What you need

In order to work with Android, the following is necessary:

- **Android SDK (latest version)** – This provides you with all of the libraries, sample code, and documentation necessary to build Android applications.
- **Java Development Kit (JDK) Version 6** – In order to run Java applications such as Eclipse, you need a Java Runtime Environment (JRE). If you want to develop Java applications, which is what you are doing when creating Android Apps, then you need a Java Development Kit, which will also include the JRE for you automatically.
- **Eclipse IDE Version 3.5** – Eclipse is a popular, open source, development environment. It will be the primary application that you use for developing your Android applications.
- **Android Development Tools** plugin for Eclipse.

Installation

Start by visiting: http://developer.android.com/sdk/index.html

The official Android SDK web site will provide you with all of the installation instructions and links necessary.

Begin by downloading the Android SDK (for your specific OS), the Java JDK (if necessary), and Eclipse IDE.

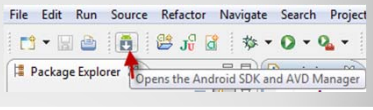
Android Virtual Device (AVD)

Once Eclipse is updated, and up and running, you can use it to develop Android applications. One of the first things that you'll want to do is set up an Android Virtual Device, or AVD for short. The AVD is a simulator of the OS as it would run on a phone. It will allow you to test your applications as you develop them.

The AVD tool allows you to simulate different types of equipment, and Android OS versions. For example, you can create an AVD that mimics Android 2.1 with an 800x480 screen resolution, or an AVD that is based on Android 1.5, and uses the 480x320 screen resolution. In fact, you should set up several AVDs, and use them all to test your applications.

The easiest way to get to the AVD tool is to click on the Android icon from within Eclipse.

This option is also available under the Window menu.



Creating an AVD

Use the Virtual Devices tab to create a new AVD.

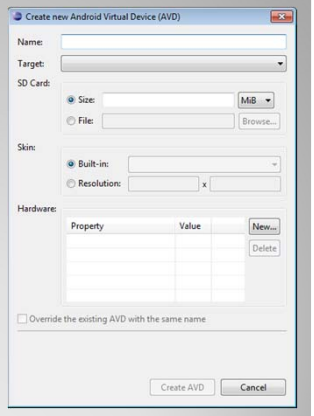
Type in a descriptive name that helps identify the OS version and screen resolution.

Choose the target (OS Version and API Version).

For **SD Card**, use at least 9mb.

Finally, choose a Built-in **Skin**, or specify your own.

Click **Create AVD** to finalize the process.



Screen Resolutions

	Low density (120), ldpi	Medium density (160), mdpi	High density (240), hdpi
Small screen	• QVGA (240x320), 2.6"-3.0" diagonal		
Normal screen	• WQVGA (240x400), 3.2"-3.5" diagonal • FWQVGA (240x432), 3.5"-3.8" diagonal	• HVGA (320x480), 3.0"-3.5" diagonal	• WVGA (480x800), 3.3"-4.0" diagonal • FWVGA (480x854), 3.5"-4.0" diagonal
Large screen		• WVGA (480x800), 4.8"-5.5" diagonal • FWVGA (480x854), 5.0"-5.8" diagonal	

http://developer.android.com/guide/practices/screens_support.html

Running the AVD

Once created, you can start your AVD. Any code that you write in Eclipse, will compile and be sent to the active AVD for testing, where you'll actually see your application run.

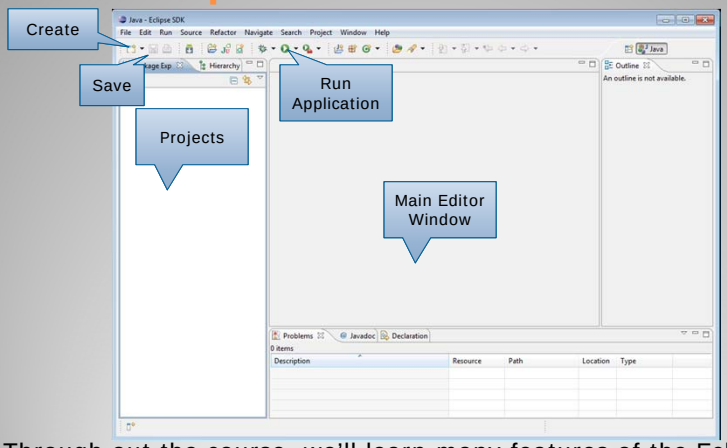
AVD

Your AVD includes a number of useful commands that can be used to help simulate different use cases for your application. For example, CTRL-F11 can switch you between portrait and landscape modes, and pressing F5 puts you into Search mode.

Emulated Device Key	Keyboard Key
Home	HOME
Menu (left softkey)	F2 or Page-up button
Star (right softkey)	Shift-F2 or Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with -trace startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

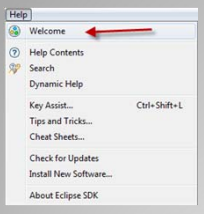
http://developer.android.com/guide/developing/tools/emulator.html

About Eclipse



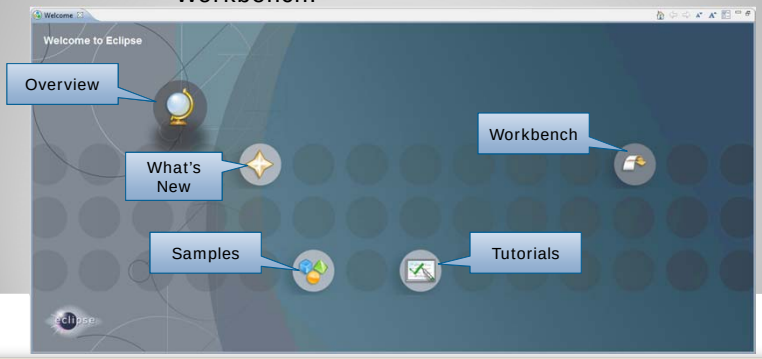
Through out the course, we'll learn many features of the Eclipse IDE, but to get us started, here are the main things to keep an eye for when working in Eclipse.

Welcome

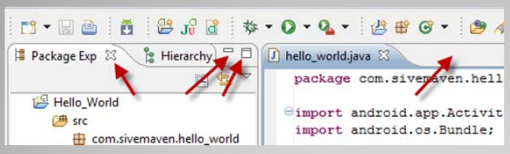


Selecting **Help → Welcome** brings up the Eclipse Welcome screen.

Here you can find on overview of features, find out what's new in the release you're working with, explore Eclipse by trying out some samples, go through tutorials, or start working in the Workbench.



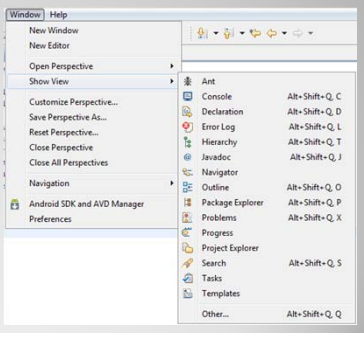
Configuring the IDE



While working in Eclipse, you can minimize, maximize, and remove panels and windows you're not working with at the time.

If you close one or more panel windows, and later need them back, use the **Window → Show View** menu to locate what you closed, and return it back to its original place.

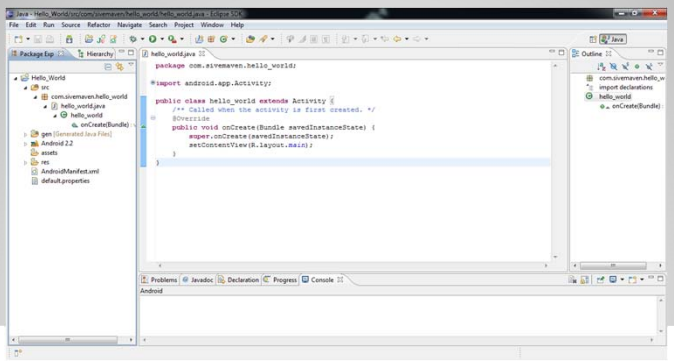
Using drag-and-drop operations, you can also move tabs from one part of the interface to another.



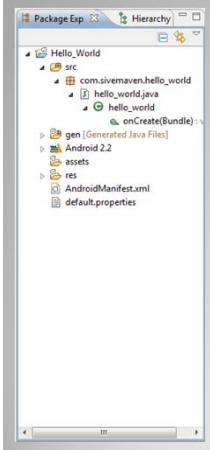
Workbench

The workbench is the main interface in which you write your code, edit various resource files, debug, and prepare releases of your applications.

The workbench can be completely customized to suit your needs. In this class, however, we will go with the default setup provided when we installed the tool.



Package Explorer



The **Package Explorer** contains a hierarchical list of all of your projects that have been worked on using Eclipse.

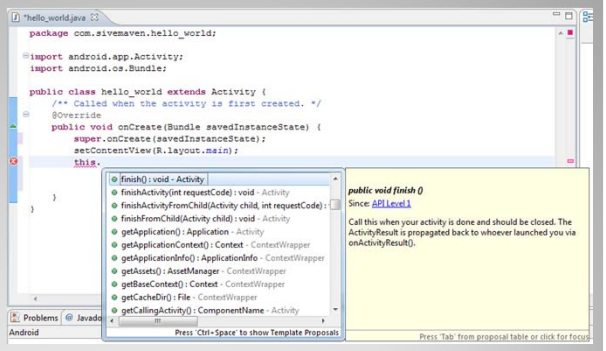
As you develop projects, you frequently need to refer back to something that you did in a different project. Because the Package Explorer is always present, it is easy to go back and forth between various code files in different projects. This becomes extremely useful as you develop more and more.

One caveat is that you need to be careful of what you're compiling and launching. Always switch your main view to the java class of the project you're working with before attempting to run it. Otherwise, you may inadvertently run a project you didn't intend to run.

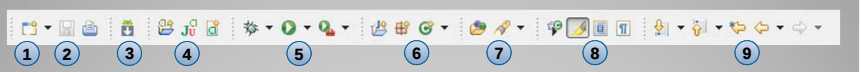
Editor Window

The editor window is where you do most of your work.

You can work with multiple documents at the same time using the tabs on top, and while writing code or editing XML files, you'll find the code hints and context sensitive drop-downs extremely useful for getting your syntax correct. Eclipse also features an automatic background compilation which will let you know when you've created a syntax error.



Menu



1. Launches the wizard for new projects or to create a new file.
2. Save – Saves the currently active editor page.
3. Android AVD Manager. Use it to launch an AVD or receive updates to the Android SDK.
4. New Android Project, New Android Testing Project, and New Android XML File.
5. Debug, Run, and External Tools.
6. New Java Project, New Java Package, and New Class File.
7. Open Type, and Search – Open Type lets you look for a particular class, and open it, while Search will look for specific text within your project files.
8. Editor formatting functions – Toggle Breadcrumbs, Mark Occurrences, Block Selection mode, and Show Whitespace.
9. General navigation tools, will let you go back and forth within a file, or re-open previous / next files that you have worked on.

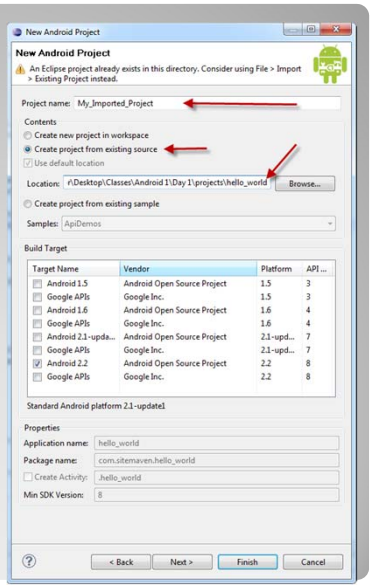
Importing Projects

One of the things you'll likely want to do at some point is import an existing project into Eclipse.

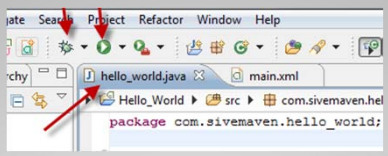
This is easy to do, but isn't immediately obvious.

Click on **File → New**, then type in a project name, select the "**Create project from existing source**" option, and choose the location of the project. Click **Finish** to bring the project into Eclipse.

Note: The location of the project will be unchanged, and it will not be in your default Workplace location.



Running Your Application

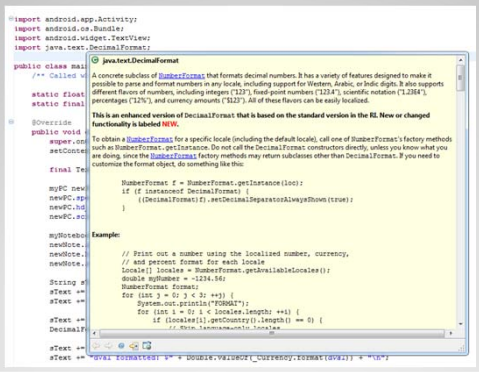


Because Eclipse lets you work in multiple projects at once, get in the good habit of selecting the correct JAVA file before attempting to run or debug your application. This will ensure that you don't accidentally run the wrong project.

It is also possible that if you attempt to run your project while viewing an XML file, that your application will not compile and error out instead. In this case, clean up the XML file error that Eclipse creates (empty text file), then use **Project → Clean** option with the correct JAVA file selected. Clean will remove cache and pre-compiled objects, and rebuild your solution from the scratch.

Using Hints

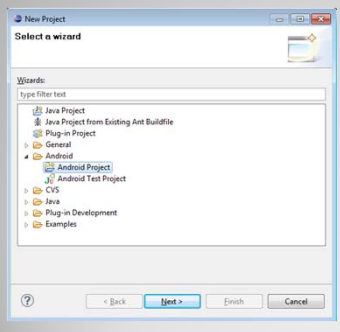
Hover over an import statement or method. When the tooltip appears, press the F2 key on the keyboard. A longer description, along with an example will appear. This hint is useful in determining what methods and functionality is available within the imported library or how to properly use a particular method.



Hello World – Creating Our Project

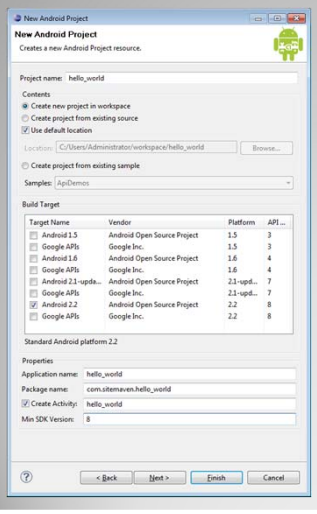
So now that everything is set up and configured, it is customary for new developers to begin their journey by creating a Hello World application.

In Eclipse, select **File**, then **New**, then **Project**.



Navigate down to the **Android** folder, then select **Android Project**, and click **Next**.

Hello World – Create Project



Next, you're going to specify the details of your project. This includes:

Project name, the version of the Android SDK to use (Build Target), the Application name, which doesn't have to be the same as Project name, name of the package, an activity (more about the package and activity later), and finally the minimum SDK version that's required for the application to run.

Hello World – Our Project files

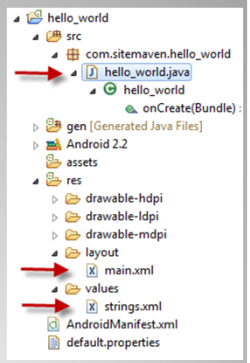
Using the Package Explorer, expand the following 3 sections, and open the files next to the arrows by double-clicking on them.

These are the main files that we are going to be looking at and working with.

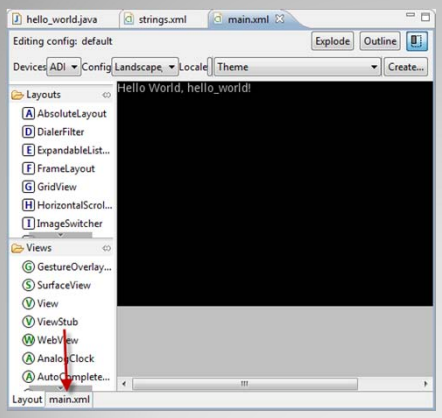
Hello_world.java is our main source code file. It contains a class called hello_world, and a create event which is trigged when our application starts. We won't be making any changes to this file today.

Main.xml contains the graphical user interface our users will work with while running our application. We'll customize that in the next step.

Finally, **strings.xml** contains the text we want to show to our users when they run our app. We'll customize that as well.



Hello World – Edit Main.xml



Switch over to main.xml, and click on the main.xml tab at the bottom of the window.

We will be editing the XML directly.

We are going to modify the textview to display our text in a larger font size, bold, and in a color other than white.

Hello World – Edit the XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" android:textColor="#00ff00" android:textSize="40px" android:textStyle="bold"/>
</LinearLayout>
```

Locate the TextView section, then add the items highlighted in yellow.

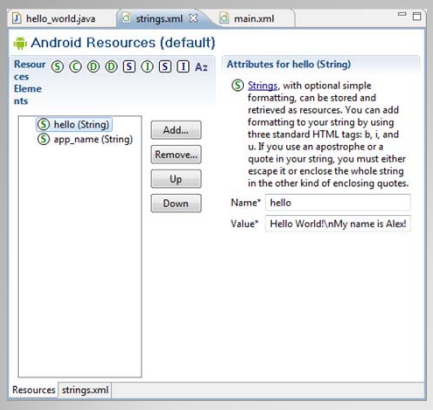
android:textColor will change the text color to red (#00ff00 is red in hex)

android:textSize will set the text size to be 40pixels.

android:textStyle will make the text bold.

Now, switch over to the Layout view to see what you did.

Hello World – Edit Strings.xml



We'll edit the strings.xml file visually. Locate the string called **hello**, then specify a new value for it:

Hello World!\nMy name is Alex!

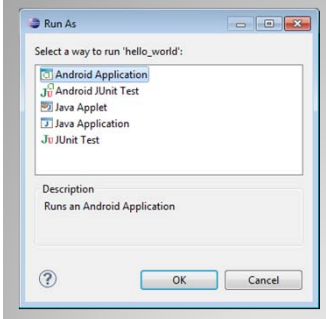
Use your name instead of mine.

Select the **app_name** string, and set it's value to:

This is my first Android project!

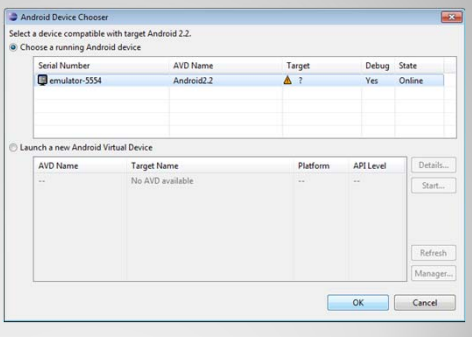
Save each of your pages (strings.xml and main.xml). Switch over to hello_world.java, then select the Run button in the main toolbar.

Hello World – Running Our Project



Select Android Application from the Run As dialog box that appears.

Next, choose the running emulator (AVD), or wait for one to start.



Hello World – App In Action!