ANDROID

application development

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Android is an open-source, Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

This tutorial will teach you the basic Android programming and will also take you through some advance concepts related to Android application development.

Audience

This tutorial has been prepared for beginners to help them understand basic Android programming. After completing this tutorial, you will find yourself at a moderate level of expertise in Android programming from where you can take yourself to next levels.

Prerequisites

Android programming is based on Java programming language. If you have a basic understanding of Java programming, then it will be fun to learn Android application development.

Copyright & Disclaimer

1. ANDROID – Overview

What is Android?

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need to develop only for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007, whereas the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Features of Android

Android is a powerful operating system competing with Apple 4GS and support great features. Few of them are listed below:

Feature	Description
Beautiful UI	Android OS basic screen provides a beautiful and intuitive user interface.
Connectivity	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
Storage	SQLite, a lightweight relational database, is used for data storage purposes.

Media support	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
Messaging	SMS and MMS
Web browser	Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
Multi-touch	Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
Multi-tasking	User can jump from one task to another and same time various application can run simultaneously.
Resizable widgets	Widgets are resizable, so users can expand them to show more content or shrink them to save space
Multi-Language	Support single direction and bi-directional text.
GCM	Google Cloud Messaging (GCM) is a service that let developers send short message data to their users on Android devices, without needing a proprietary sync solution.
Wi-Fi Direct	A technology that let apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
Android Beam	A popular NFC-based technology that let users instantly share, just by touching two NFC-enabled phones together.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play** or the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and is growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

2. ANDROID – Environment Setup

You will be glad to know that you can start your Android application development on either of the following operating systems:

- Microsoft Windows XP or later version.
- Mac OS X 10.5.8 or later version with Intel chip.
- Linux including GNU C Library 2.7 or later.

Second point is that all the required tools to develop Android applications are freely available and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

- Java JDK5 or JDK6
- Android SDK
- Eclipse IDE for Java Developers (optional)
- Android Development Tools (ADT) Eclipse Plugin (optional)

Here last two components are optional and if you are working on Windows machine then these components make your life easy while doing Java based application development. So let us have a look at how to proceed to set the required environment.

Step 1 - Setup Java Development Kit (JDK)

You can download the latest version of Java JDK from Oracle's Java site: Java SE Downloads. You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally, set PATH and JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, typically java_install_dir/bin and java_install_dir respectively.

If you are running Windows and have installed the JDK in C:\jdk1.6.0_15, you would have to put the following line in your C:\autoexec.bat file.

```
set PATH=C:\jdk1.6.0_15\bin;%PATH%
set JAVA_HOME=C:\jdk1.6.0_15
```

Alternatively, you could also right-click on *My Computer*, select *Properties*, then *Advanced*, then *Environment Variables*. Then, you would update the PATH value and press the OK button.

On Linux, if the SDK is installed in /usr/local/jdk1.6.0_15 and you use the C shell, you would put the following code into your **.cshrc** file.

```
setenv PATH /usr/local/jdk1.6.0_15/bin:$PATH
setenv JAVA_HOME /usr/local/jdk1.6.0_15
```

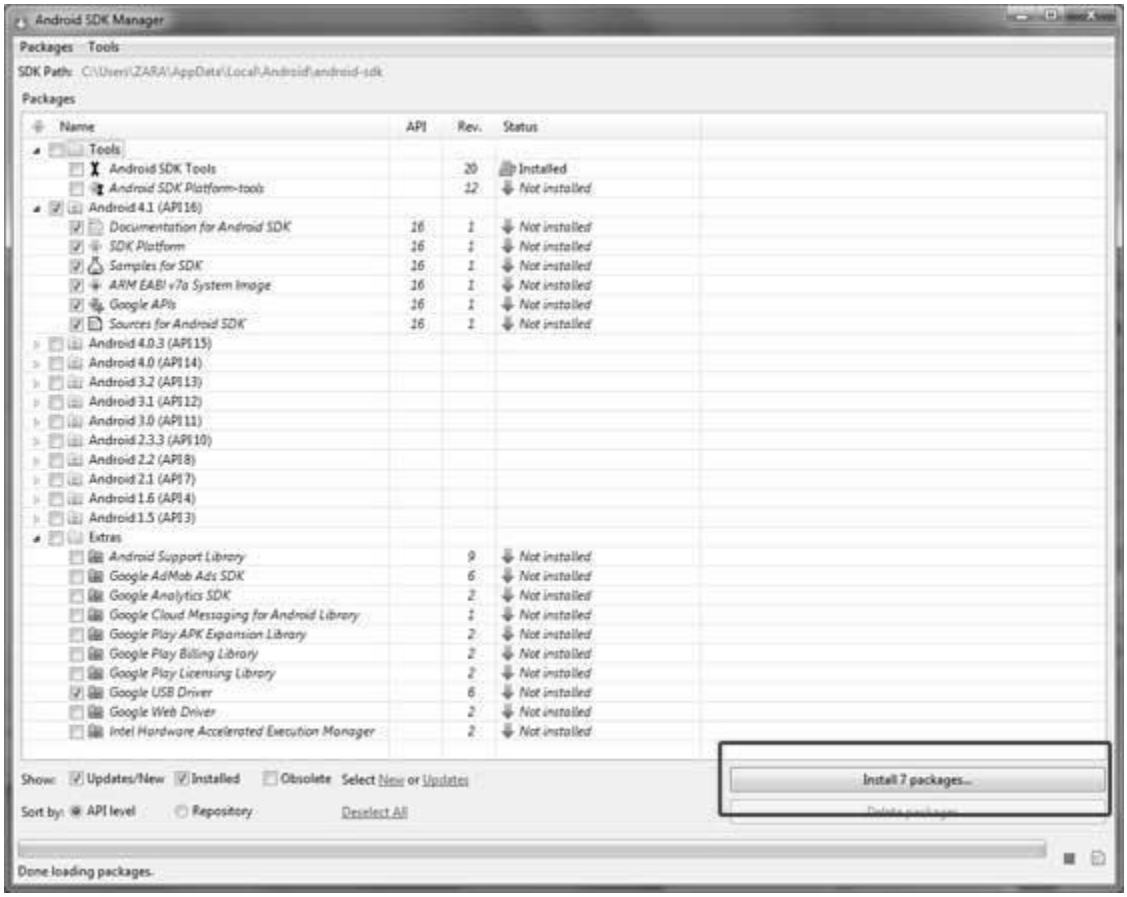
Alternatively, if you use an Integrated Development Environment (IDE) Eclipse, then it will know automatically where you have installed your Java.

Step 2 - Setup Android SDK

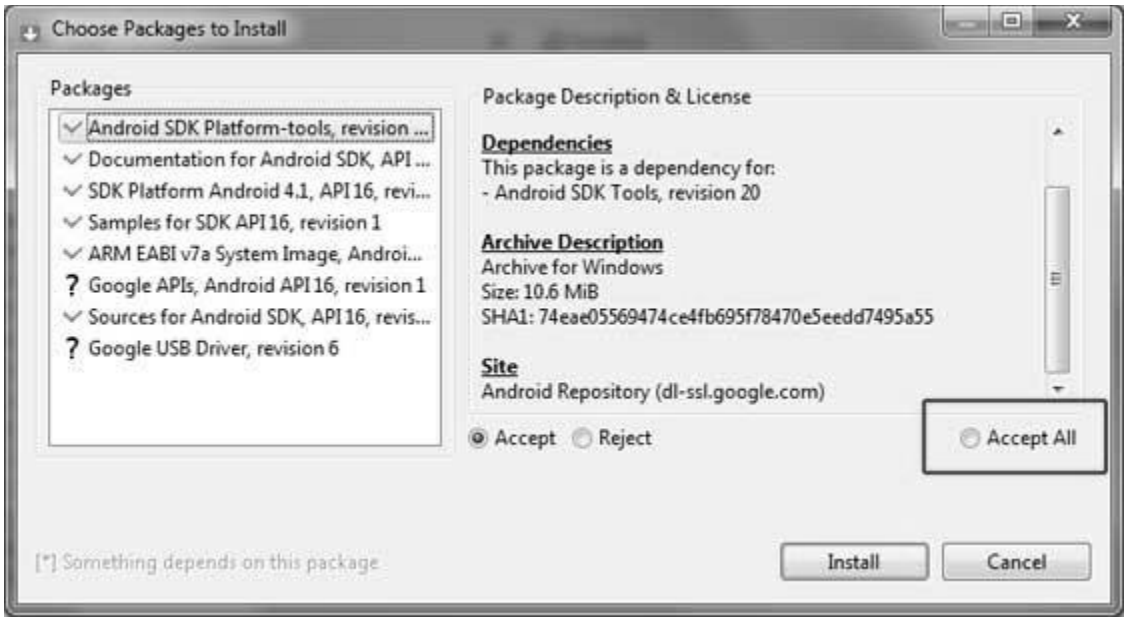
You can download the latest version of Android SDK from Android's official website: http://developer.android.com/sdk/index.html. If you are installing SDK on Windows machine, then you will find a*installer_rXX-windows.exe*, so just download and run this exe which will launch *Android SDK Tool Setup* wizard to guide you throughout the installation, so just follow the instructions carefully. Finally, you will have *Android SDK Tools* installed on your machine.

If you are installing SDK either on Mac OS or Linux, check the instructions provided along with the downloaded *android-sdk_rXX-macosx.zip* file for Mac OS and *android-sdk_rXX-linux.tgz* file for Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 7 operating system.

So let's launch *Android SDK Manager* using the option **All Programs > Android SDK Tools > SDK Manager**, this will give you following window:



Once you launched SDK manager, it is time to install other required packages. By default it will list down total 7 packages to be installed, but we will suggest to de-select *Documentation for Android SDK* and *Samples for SDK* packages to reduce installation time. Next click the **Install 7 Packages** button to proceed, which will display following dialogue box:



If you agree to install all the packages, select **Accept All** radio button and proceed by clicking **Install** button. Now let SDK manager do its work and you go, pick up a cup of coffee and wait until all the packages are installed. It may take some time depending on your internet connection. Once all the packages are installed, you can close SDK manager using top-right cross button.

Step 3 - Setup Eclipse IDE

All the examples in this tutorial have been written using Eclipse IDE. So we would suggest you should have latest version of Eclipse installed on your machine.

To install Eclipse IDE, download the latest Eclipse binaries from http://www.eclipse.org/downloads/. Once you have downloaded the installation, unpack the binary distribution into a convenient location. For example in C:\eclipse on windows, or /usr/local/eclipse on Linux and finally set PATH variable appropriately.

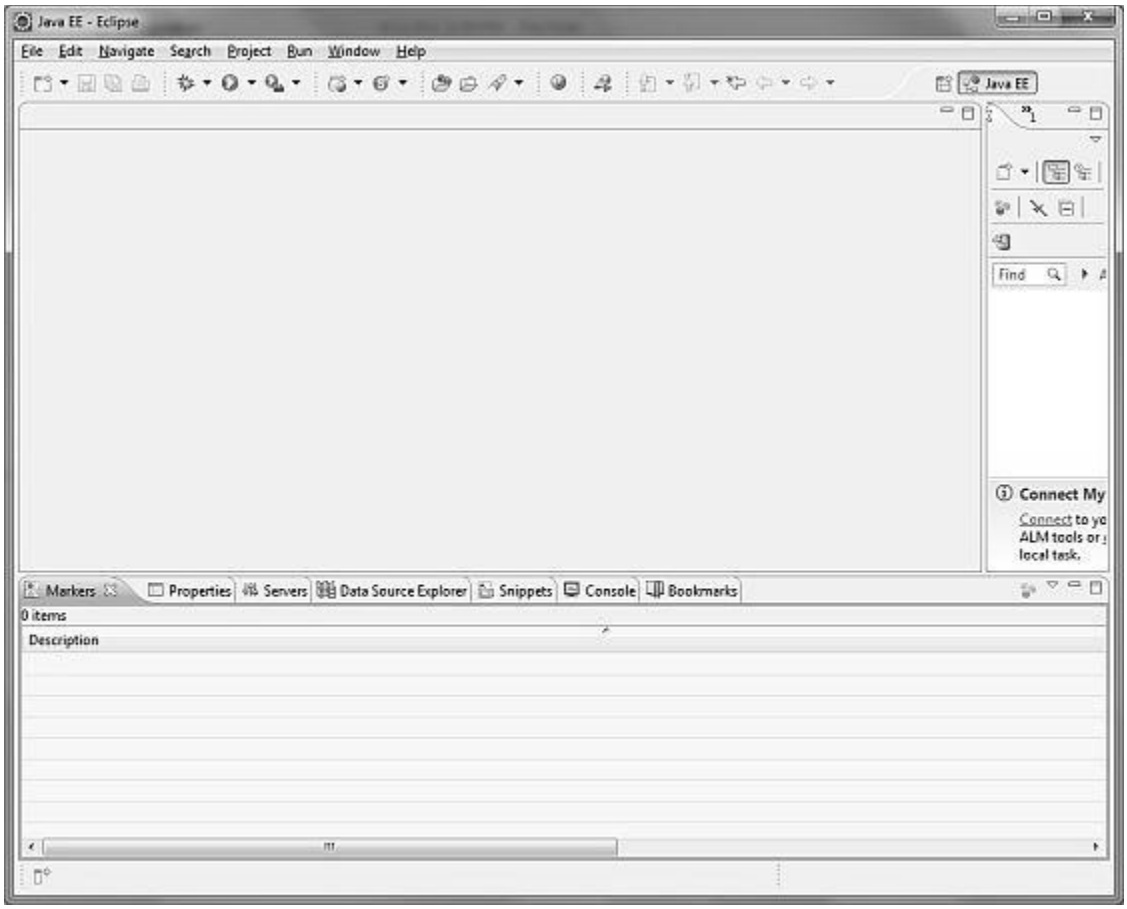
Eclipse can be started by executing the following commands on windows machine, or you can simply double click on eclipse.exe

```
%C:\eclipse\eclipse.exe
```

Eclipse can be started by executing the following command on Linux machine:

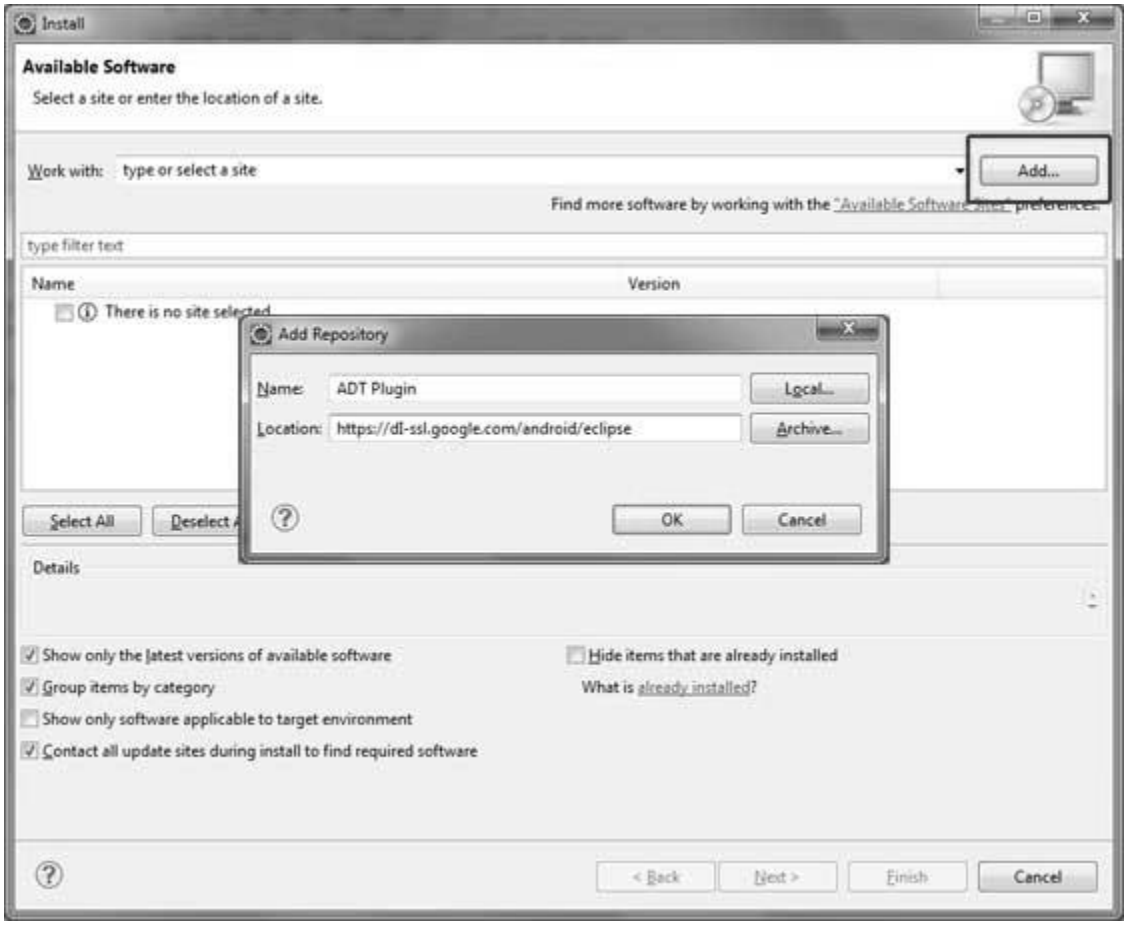
```
$/usr/local/eclipse/eclipse
```

After a successful startup, if everything is fine then it should display the following result:

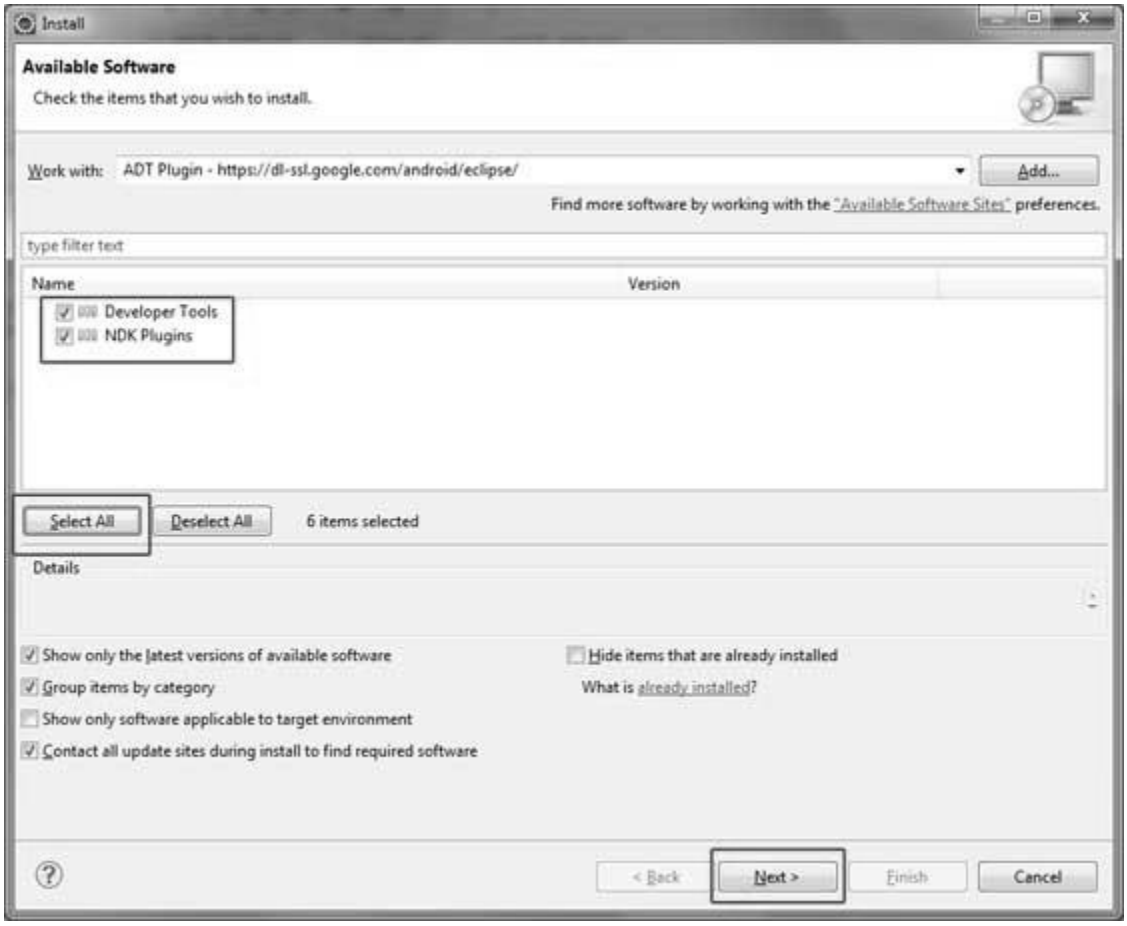


Step 4 - Setup Android Development Tools (ADT) Plugin

This step will help you in setting Android Development Tool plugin for Eclipse. Let's start with launching Eclipse and then, choose **Help > Software Updates > Install New Software**. This will display the following dialogue box.



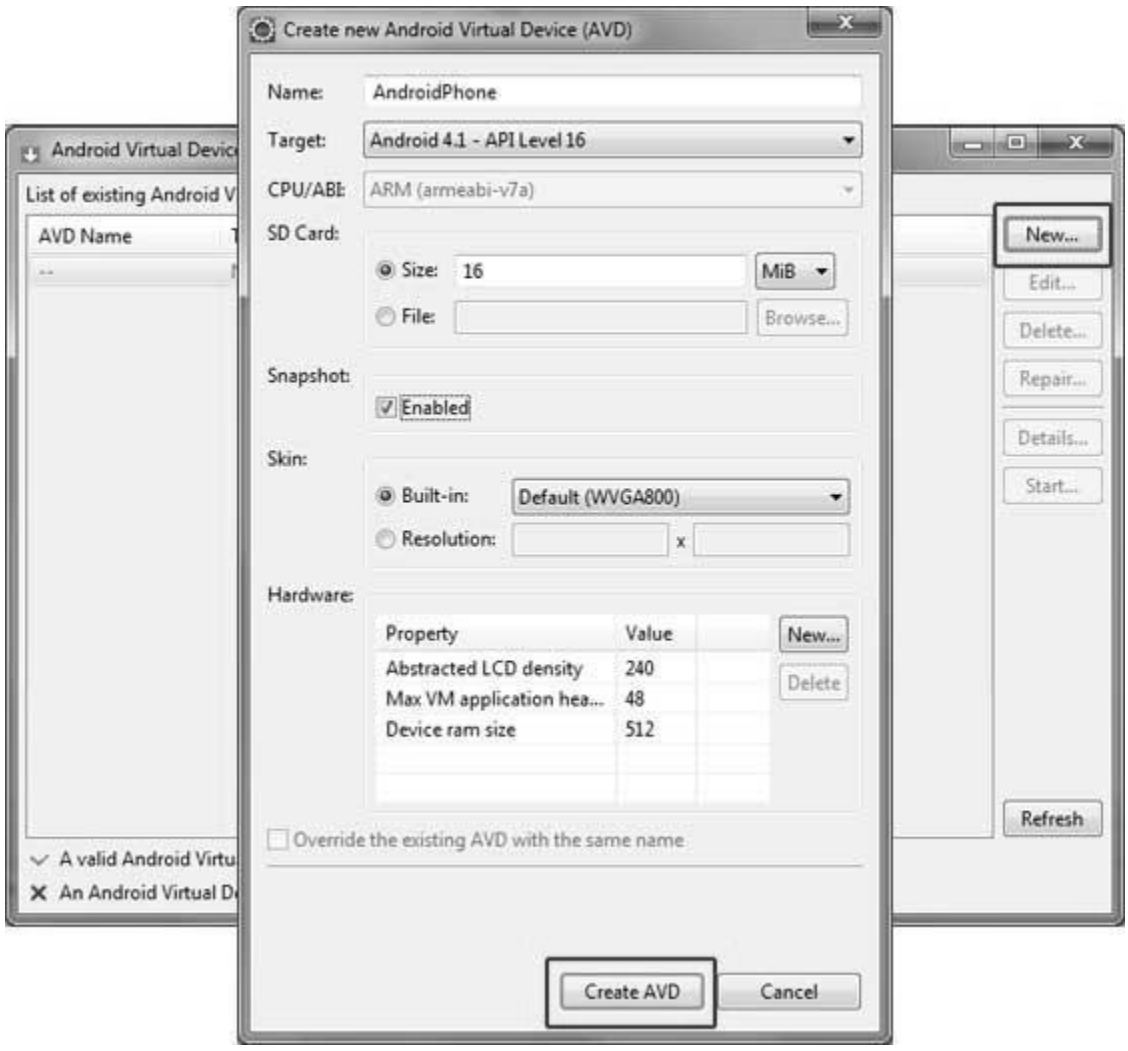
Now use **Add** button to add *ADT Plugin* as name and *https://dl-ssl.google.com/android/eclipse/* as the location. Then click OK to add this location. As soon as you will click OK button to add this location, Eclipse starts searching for the plug-in available in the given location and finally lists down the found plugins.



Now select all the listed plug-ins using **Select All** button and click **Next** button which will guide you ahead to install Android Development Tools and other required plugins.

Step 5 - Create Android Virtual Device

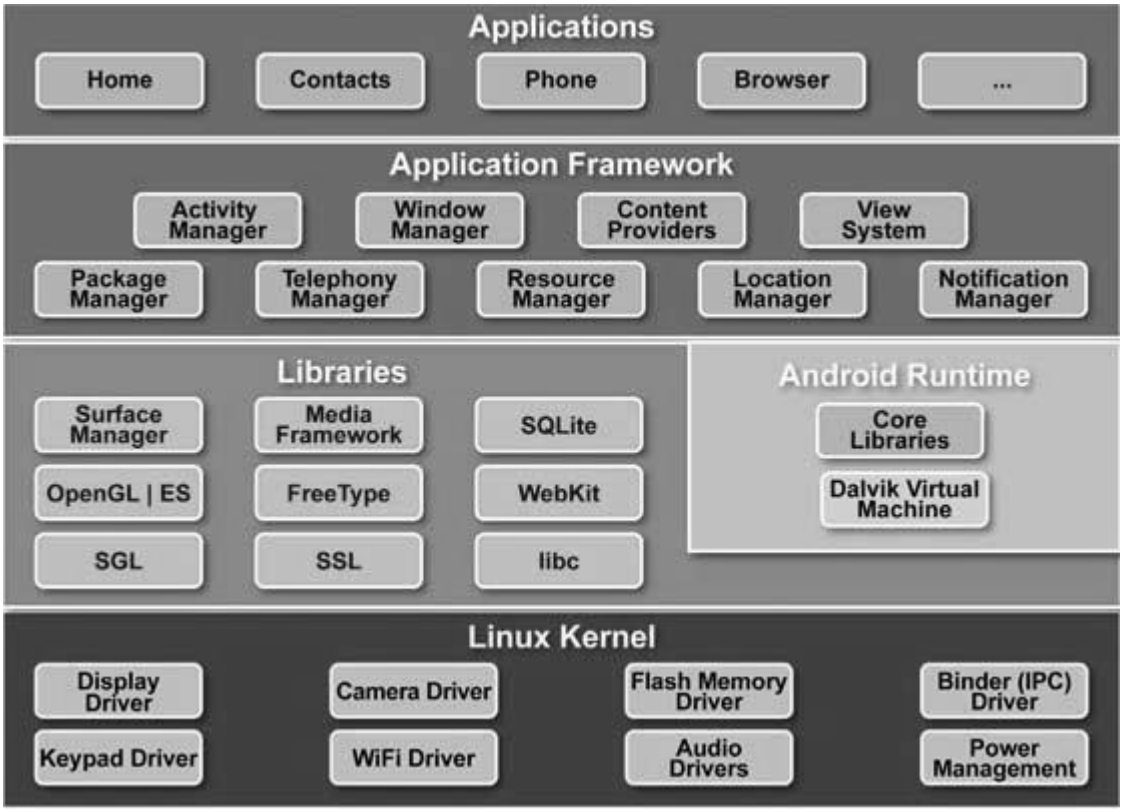
To test your Android applications you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager using Eclipse menu options **Window > AVD Manager>** which will launch Android AVD Manager. Use **New** button to create a new Android Virtual Device and enter the following information, before clicking **Create AVD** button.



If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re-start your machine and once you are done with this last step, you are ready to proceed for your first Android example but before that we will see few more important concepts related to Android Application Development.

3. ANDROID – Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at, such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and

sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games, etc.

4. ANDROID – Applications Component

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

Components	Description
Activities	They dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Activities

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and one for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows:

```
public class MainActivity extends Activity
{


}
```

Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows:

```
public class MyService extends Service
{


}
```

Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcasted as an **Intent** object.

```
public class MyReceiver  extends  BroadcastReceiver
{


}
```

Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends  ContentProvider
{
```

```
}
```

We will go through these tags in detail while covering application components in individual chapters.

Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are:

Components	Description
Fragments	Represent a behavior or a portion of user interface in an Activity.
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resources	External elements, such as strings, constants and drawable pictures.
Manifest	Configuration file for the application.

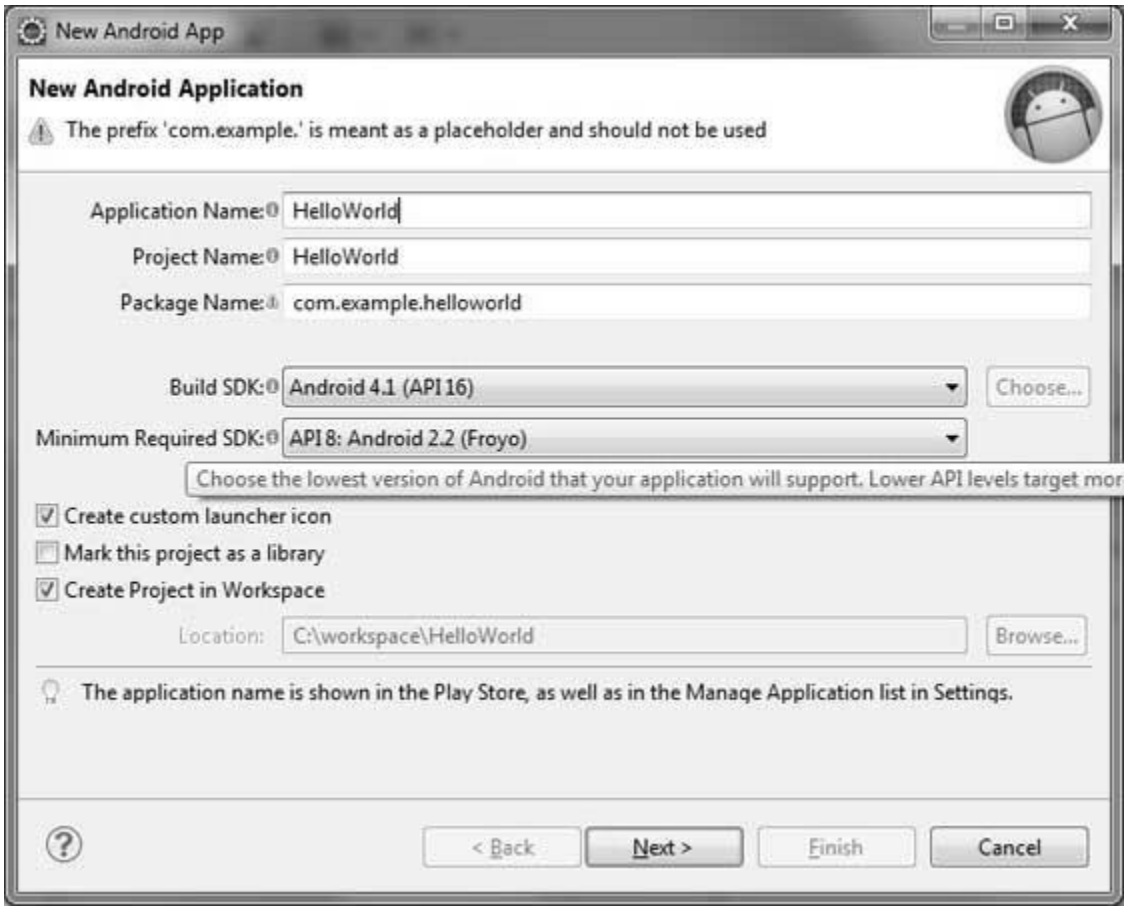
5. ANDROID – Hello World Example

Let us start actual programming with Android Framework. Before you start writing your first example using Android SDK, you have to make sure that you have setup your Android development environment properly as explained in Android - Environment Setup tutorial. We also assume, that you have a little bit working knowledge with Eclipse IDE.

So let us proceed to write a simple Android Application which will print "Hello World!".

Create Android Application

The first step is to create a simple Android Application using Eclipse IDE. Follow the option **File -> New -> Project** and finally select **Android New Application** wizard from the wizard list. Now name your application as **HelloWorld** using the wizard window as follows:



	This contains the **.java** source files for your project. By default, it includes an*MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.
2	**gen** This contains the **.R** file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
3	**bin** This folder contains the Android package files **.apk** built by the ADT during the build process and everything else needed to run an Android application.
4	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
5	**res/layout** This is a directory for files that define your app's user interface.
6	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.
7	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

Following section will give a brief overview few of the important application files.

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application:

```
package com.example.helloworld;
```

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are fired when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest* file called **AndroidManifest.xml** which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.helloworld"
   android:versionCode="1"
   android:versionName="1.0" >
```

```
   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="15" />
   <application
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >
       <activity
           android:name=".MainActivity"
           android:label="@string/title_activity_main" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category
            android:name="android.intent.category.LAUNCHER"/>
           </intent-filter>
       </activity>
   </application>
</manifest>
```

Here <application>...</application> tags enclosed the components related to the application. Attribute *android:icon* will point to the application icon available under*res/drawable-hdpi*. The application uses the image named ic_launcher.png located in the drawable folders.

The <activity> tag is used to specify an activity and *android:name* attribute specifies the fully qualified class name of the *Activity* subclass and the *android:label* attributes specifies a string to use as the label for the activity. You can specify multiple activities using <activity> tags.

The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.

The *@string* refers to the *strings.xml* file explained below. Hence, *@string/app_name* refers to the *app_name* string defined in the strings.xml file, which is "HelloWorld". Similar way, other strings get populated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components: