ANDROID STUDIO

INTRODUCTION

***Android Studio (Basics)***

**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 only develop 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 where as 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.

**Why Android ?**

**Features of Android**

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

|  |  |
| --- | --- |
| **Sr.No.** | **Feature & Description** |
| 1 | **Beautiful UI**  Android OS basic screen provides a beautiful and intuitive user interface. |
| 2 | **Connectivity**  GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX. |
| 3 | **Storage**  SQLite, a lightweight relational database, is used for data storage purposes. |
| 4 | **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. |
| 5 | **Messaging**  SMS and MMS |
| 6 | **Web browser**  Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3. |
| 7 | **Multi-touch**  Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. |
| 8 | **Multi-tasking**  User can jump from one task to another and same time various application can run simultaneously. |
| 9 | **Resizable widgets**  Widgets are resizable, so users can expand them to show more content or shrink them to save space. |
| 10 | **Multi-Language**  Supports single direction and bi-directional text. |
| 11 | **GCM**  Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution. |
| 12 | **Wi-Fi Direct**  A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection. |
| 13 | **Android Beam**  A popular NFC-based technology that lets 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**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and 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 growing fast. Every day more than 1 million new Android devices are activated worldwide.

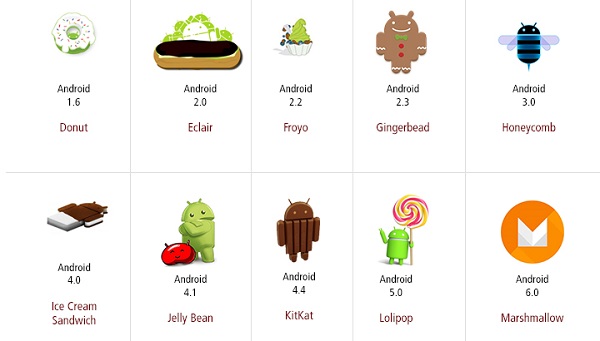
This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

**Categories of Android applications**

There are many android applications in the market. The top categories are −

**History of Android**

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.



**What is API level?**

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform Version** | **API Level** | **VERSION\_CODE** |  |
| Android 6.0 | 23 | MARSHMALLOW |  |
| Android 5.1 | 22 | LOLLIPOP\_MR1 |  |
| Android 5.0 | 21 | LOLLIPOP |  |
| Android 4.4W | 20 | KITKAT\_WATCH | KitKat for Wearables Only |
| Android 4.4 | 19 | KITKAT |  |
| Android 4.3 | 18 | JELLY\_BEAN\_MR2 |  |
| Android 4.2, 4.2.2 | 17 | JELLY\_BEAN\_MR1 |  |
| Android 4.1, 4.1.1 | 16 | JELLY\_BEAN |  |
| Android 4.0.3, 4.0.4 | 15 | ICE\_CREAM\_SANDWICH\_MR1 |  |
| Android 4.0, 4.0.1, 4.0.2 | 14 | ICE\_CREAM\_SANDWICH |  |
| Android 3.2 | 13 | HONEYCOMB\_MR2 |  |
| Android 3.1.x | 12 | HONEYCOMB\_MR1 |  |
| Android 3.0.x | 11 | HONEYCOMB |  |
| Android 2.3.4  Android 2.3.3 | 10 | GINGERBREAD\_MR1 |  |
| Android 2.3.2  Android 2.3.1  Android 2.3 | 9 | GINGERBREAD |  |
| Android 2.2.x | 8 | FROYO |  |
| Android 2.1.x | 7 | ECLAIR\_MR1 |  |
| Android 2.0.1 | 6 | ECLAIR\_0\_1 |  |
| Android 2.0 | 5 | ECLAIR |  |
| Android 1.6 | 4 | DONUT |  |
| Android 1.5 | 3 | CUPCAKE |  |
| Android 1.1 | 2 | BASE\_1\_1 |  |
| Android 1.0 | 1 | BASE |  |

**ENVIRONMENT SETTINGS**



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 later version
* Android Studio

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 how to proceed to set required environment.

**Set-up Java Development Kit (JDK)**

You can download the latest version of Java JDK from Oracle's Java site − [Java SE Downloads](http://www.oracle.com/technetwork/java/javase/downloads/index.html). 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 installed the JDK in C:\jdk1.8.0\_102, you would have to put the following line in your C:\autoexec.bat file.

set PATH=C:\jdk1.8.0\_102\bin;%PATH%

set JAVA\_HOME=C:\jdk1.8.0\_102

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/jdk1.8.0\_102 and you use the C shell, you would put the following code into your **.cshrc** file.

setenv PATH /usr/local/jdk1.8.0\_102/bin:$PATH

setenv JAVA\_HOME /usr/local/jdk1.8.0\_102

Alternatively, if you use Android studio, then it will know automatically where you have installed your Java.

**Android IDEs**

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows

* [Android Studio](https://www.tutorialspoint.com/android/android_studio.htm)
* [Eclipse IDE(Deprecated)](https://www.tutorialspoint.com/android/android_eclipse.htm)

# 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 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers 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 Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

* **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
* **android.content** − Facilitates content access, publishing and messaging between applications and application components.
* **android.database** − Used to access data published by content providers and includes SQLite database management classes.
* **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
* **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
* **android.text** − Used to render and manipulate text on a device display.
* **android.view** − The fundamental building blocks of application user interfaces.
* **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
* **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

## 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.

The Android framework includes the following key services −

* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notifications Manager** − Allows applications to display alerts and notifications to the user.
* **View System** − An extensible set of views used to create application user interfaces.

## 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.

# Android - Application Components

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 −

|  |  |
| --- | --- |
| **Sr.No** | **Components & Description** |
| 1 | **Activities**  They dictate the UI and handle the user interaction to the smart phone screen. |
| 2 | **Services**  They handle background processing associated with an application. |
| 3 | **Broadcast Receivers**  They handle communication between Android OS and applications. |
| 4 | **Content Providers**  They handle data and database management issues. |

**Activities**

An activity represents a single screen with a user interface,in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity 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 broadcaster as an **Intent** object.

public class MyReceiver extends BroadcastReceiver {

public void onReceive(context,intent){}

}

**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 {

public void onCreate(){}

}

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 −

|  |  |
| --- | --- |
| **S.No** | **Components & Description** |
| 1 | **Fragments**  Represents a portion of user interface in an Activity. |
| 2 | **Views**  UI elements that are drawn on-screen including buttons, lists forms etc. |
| 3 | **Layouts**  View hierarchies that control screen format and appearance of the views. |
| 4 | **Intents**  Messages wiring components together. |
| 5 | **Resources**  External elements, such as strings, constants and drawable pictures. |
| 6 | **Manifest**  Configuration file for the application. |

# 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 set-up your Android development environment properly as explained in [Android - Environment Set-up](https://www.tutorialspoint.com/android/android_environment_setup.htm) tutorial. I also assume that you have a little bit working knowledge with Android studio.

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 Android studio. When you click on Android studio icon, it will show screen as shown below