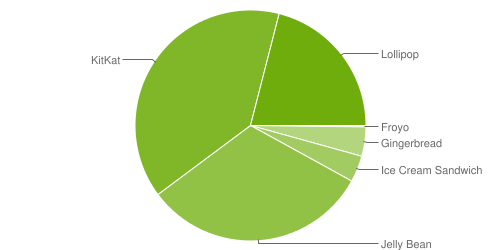
## **Platform Versions**

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read [Supporting Different Platform Versions](https://developer.android.com/training/basics/supporting-devices/platforms.html).

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Codename | API | Distribution |
| [2.2](https://developer.android.com/about/versions/android-2.2.html) | Froyo | 8 | 0.2% |
| [2.3.3 - 2.3.7](https://developer.android.com/about/versions/android-2.3.3.html) | Gingerbread | 10 | 4.1% |
| [4.0.3 - 4.0.4](https://developer.android.com/about/versions/android-4.0.html) | Ice Cream Sandwich | 15 | 3.7% |
| [4.1.x](https://developer.android.com/about/versions/android-4.1.html) | Jelly Bean | 16 | 12.1% |
| [4.2.x](https://developer.android.com/about/versions/android-4.2.html) | 17 | 15.2% |
| [4.3](https://developer.android.com/about/versions/android-4.3.html) | 18 | 4.5% |
| [4.4](https://developer.android.com/about/versions/android-4.4.html) | KitKat | 19 | 39.2% |
| [5.0](https://developer.android.com/about/versions/android-5.0.html) | Lollipop | 21 | 15.9% |
| [5.1](https://developer.android.com/about/versions/android-5.1.html) | 22 | 5.1% |



Data collected during a 7-day period ending on September 7, 2015.Any versions with less than 0.1% distribution are not shown.

## **History of Android**

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

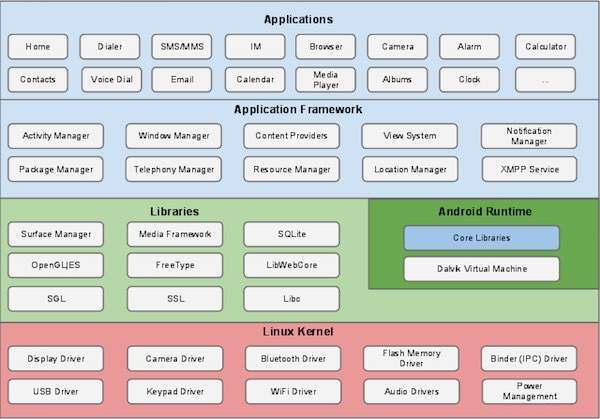


## **What is API level?**

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 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 |  |

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 −

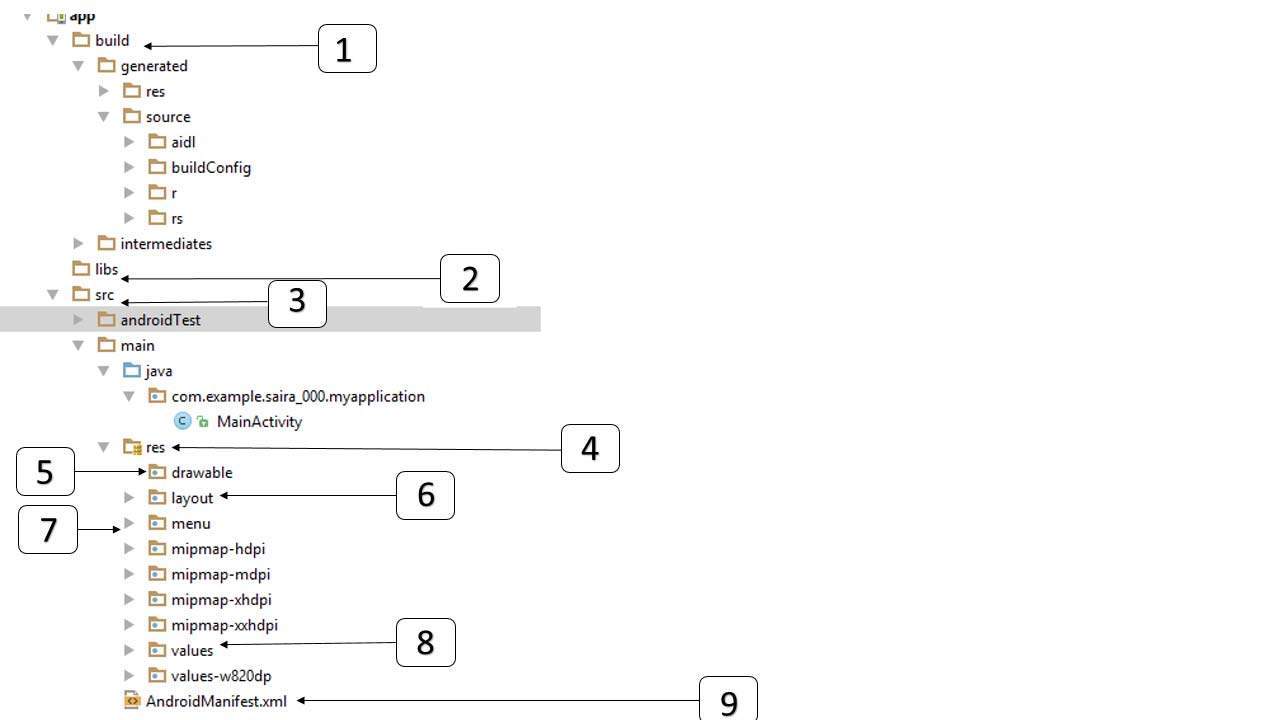
* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user 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.

## **Anatomy of Android Application**

You should be aware of a few directories and files in an Android project:



|  |  |
| --- | --- |
| **Sr.No** | **Folder, File & Description** |
| 1 | **build**  This contains the auto generated file which are as Aidl,Build configuration, and R(R.JAVA) |
| 2 | **Libs**  This is a directory to add the libraries to develop the android applications |
| 3 | **src**  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. |
| 4 | **res**  This is a directory,which is having drawable,layout,values,and android manifest file |
| 5 | **res/drawable-hdpi**  This is a directory for drawable objects that are designed for high-density screens. |
| 6 | **res/layout**  This is a directory for files that define your app's user interface. |
| 7 | **res/menu**  This is a directory for menu objects that are designed to make menu in android applications |
| 8 | **res/values**  This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions. |
| 9 | **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 fi red 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.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="22" />

<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 fi le, 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 −

* <activity>elements for activities
* <service> elements for services
* <receiver> elements for broadcast receivers
* <provider> elements for content providers

## **The Strings File**

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file −

<resources>

<string name="app\_name">HelloWorld</string>

<string name="hello\_world">Hello world!</string>

<string name="menu\_settings">Settings</string>

<string name="title\_activity\_main">MainActivity</string>

</resources>

## **The Layout File**

The **activity\_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout −

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:padding="@dimen/padding\_medium"

android:text="@string/hello\_world"

tools:context=".MainActivity" />

</RelativeLayout>

This is an example of simple *RelativeLayout* which we will study in a separate chapter. The *TextView* is an Android control used to build the GUI and it have various attribuites like *android:layout\_width*, *android:layout\_height* etc which are being used to set its width and height etc. The *@string* refers to the strings.xml file located in the res/values folder. Hence, @string/hello\_world refers to the hello string defined in the strings.xml fi le, which is "Hello World!".