MOBILE APP DEVELOPMENT

Introduction to Android

Introduction to 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.

Features of Android

- Android Open Source Project so we can customize the OS based on our requirements.
- Android supports different types of connectivity for GSM, CDMA, Wi-Fi, Bluetooth, etc. for telephonic conversation or data transfer.
- Using wifi technology we can pair with other devices while playing games or using other applications.
- It contains multiple APIs to support location-tracking services such as GPS.
- We can manage all data storage-related activities by using the file manager.
- It contains a wide range of media supports like AVI, MKV, FLV, MPEG4, etc. to play or record a variety of audio/video.
- It also supports different image formats like JPEG, PNG, GIF, BMP, MP3, etc.
- It supports multimedia hardware control to perform playback or recording using a camera and microphone.
- Android has an integrated open-source WebKit layout-based web browser to support User Interfaces like HTML5, and CSS3.
- Android supports multi-tasking means we can run multiple applications at a time and can switch between them.
- It provides support for virtual reality or 2D/3D Graphics.

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.

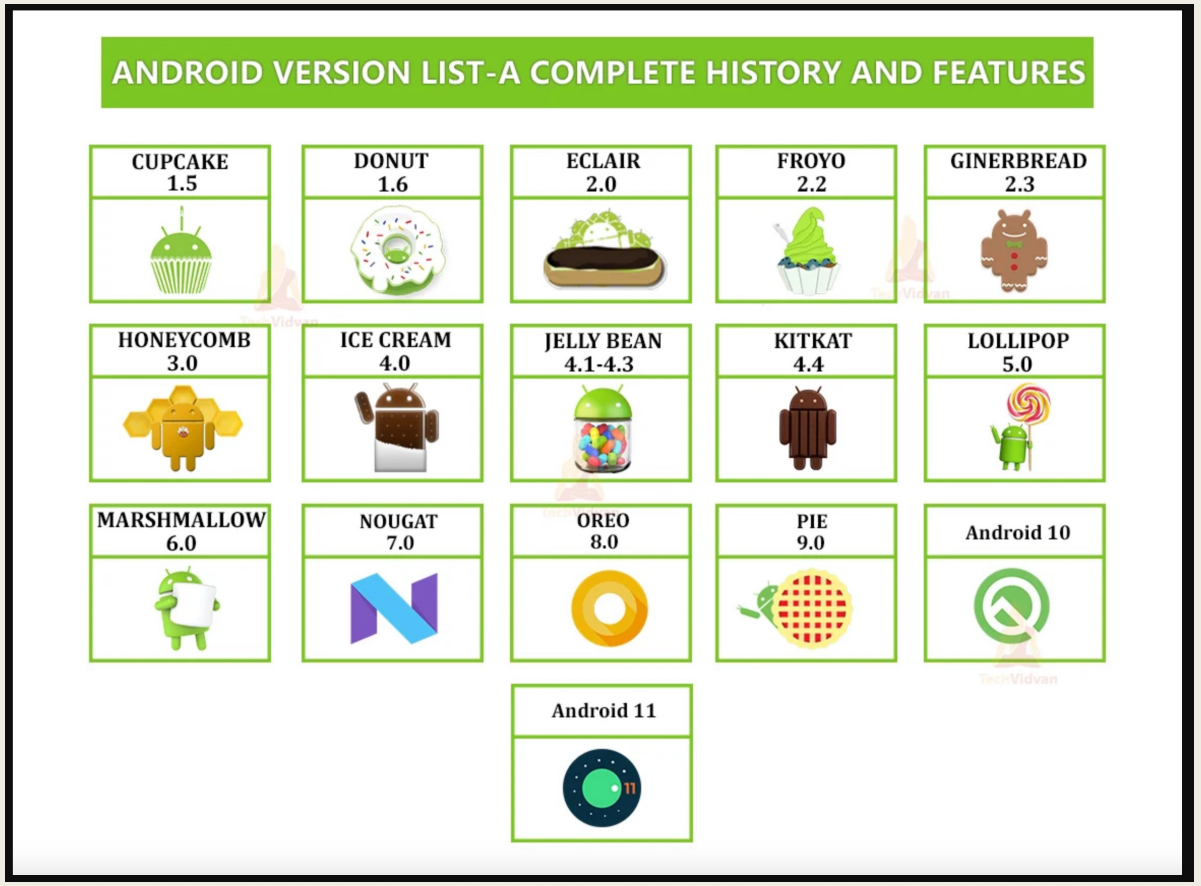
This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

Android Versions

Name	Version	API	Date Release	Security Fixes?
No codename	1.0	1	September 23, 2008	No
No codename	1.1	2	February 9, 2009	No
Cupcake	1.5	3	April 27, 2009	No
Donut	1.6	4	September 15, 2009	No
Eclair	2.0 – 2.1	5 – 7	October 26, 2009	No
Froyo	2.2 – 2.2.3	8	May 20, 2010	No
Gingerbread	2.3 – 2.3.7	9 – 10	December 6, 2010	No
Honeycomb	3.0 – 3.2.6	11 – 13	February 22, 2011	No
Ice Cream Sandwich	4.0 – 4.0.4	14 – 15	October 18, 2011	No

Android Versions

Jelly Bean	4.1 – 4.3.1	16 – 18	July 9, 2012	No
KitKat	4.4 – 4.4.4	19 – 20	October 31, 2013	No
Lollipop	5.0 – 5.1.1	21- 22	November 12, 2014	No
Marshmallow	6.0 – 6.0.1	23	October 5, 2015	No
Nougat	7.0	24	August 22, 2016	No
Nougat	7.1.0 – 7.1.2	25	October 4, 2016	No
Oreo	8.0	26	August 21, 2017	No
Oreo	8.1	27	December 5, 2017	Yes
Pie	9.0	28	August 6, 2018	Yes
Android 10	10.0	29	September 3, 2019	Yes
Android 11	11	30	September 8, 2020	Yes
Android 12	12	31	To Be Announced	Pre Supported



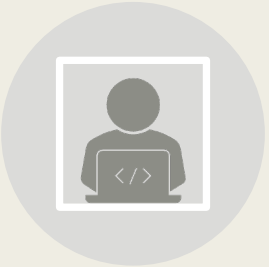
ANDROID VERSIONS

Prerequisites of Android

Java/Kotlin: You should have a basic knowledge of either Java or Kotlin.

Database: If you have some knowledge of SQL and SQLite, then it's pretty good.

XML: A basic knowledge of XML is required.

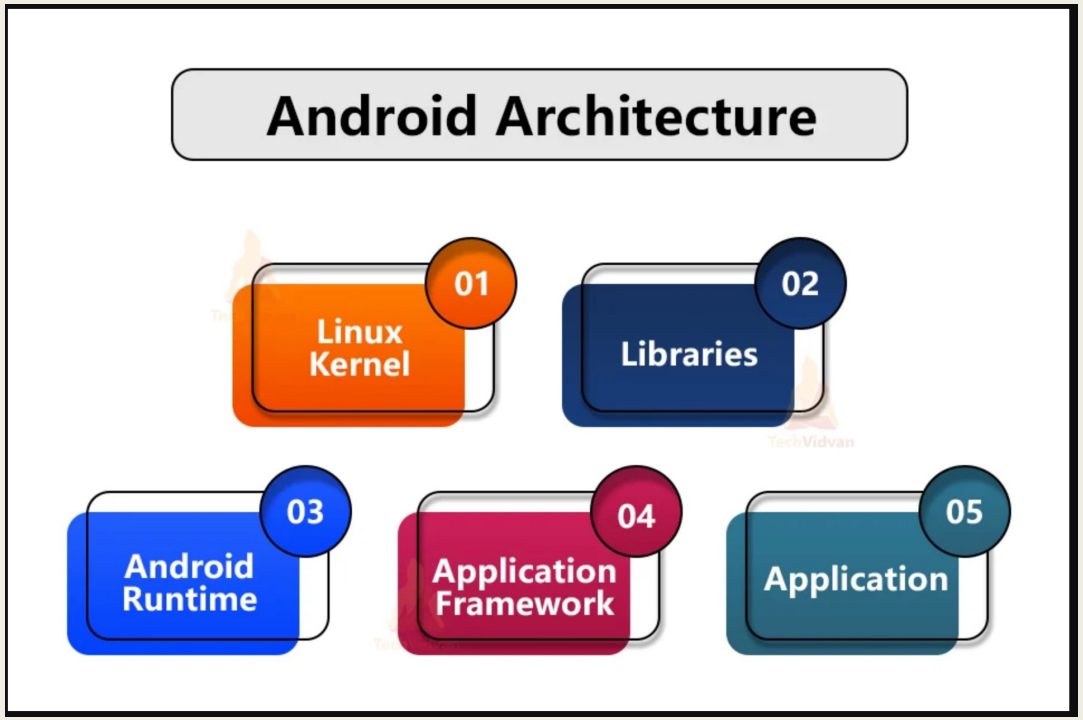


OOP: Knowing the Object-Oriented Programming(OOP's) concepts would be an add-on and help you during app development.

Android Architecture

Any Operating System has its architecture to carry on various functionalities. Similarly, android has its architecture. The android architecture gives us an idea about its design and the build. The architecture can be subdivided into further categories as below:

- Linux Kernel
- Libraries
- Android Runtime
- Application Framework
- Applications



Android Architecture

Linux Kernel: Linux Kernel carries all the drivers for the low-level devices like Audio Driver, Wi-Fi Driver, Flash Memory Driver, Bluetooth driver, Camera Driver, Keypad Driver, etc. It is also the abstract layer of android.

Libraries: There are several libraries to provide various functionality for purposes like android development. These libraries are written in C/C++ and are an essential part of architecture.

Android Runtime: It provides us with an environment for executing and debugging our android applications.

Android Architecture

Application Framework: It has many packages in it which are implemented in Java.

Applications: At the top level of the architecture, you have the applications. Applications can be of system or user or even kept by the OEM manufacturers. Phone, Messaging, Camera, Gallery, etc., are some of the standard applications present in any android device.

Advantages of Android

There are a lot of advantages of Android, and the following are some of them:

- Community Support: Google being the developer and supporter of Android, always provides necessary community support and has discussion forums to clear issues.
- Predefined Layouts and widgets: Google's Android Studio provides features like predefined layouts and widgets, making our task more manageable.
- Fragment Support: Android has fragments using which we need not create multiple activities. Using one single activity, we can perform several tasks. One such example is Whatsapp, where we see a single activity containing a trio of Chats, Status, and Calls.

Advantages of Android

- Broad Segment of Users: In the whole world, we know that most often, people have an android device.
- Easy Publishing: You can publish any of your developed apps on the Google Play Store and start your venture.
- Immense Support: Almost all devices can support Android as an operating system.
- Multitasking: Android enables multitasking for the users.

Disadvantages of Android

- Low Security: Often, we see android apps getting breached or hacked and people's data getting leaked. However, at its best, Google is always coming up with security patches and resolving such issues.
- Testing: Sometimes, testing an android app is ridiculous as its performance or state may vary from device to device and even at different Android versions.
- Power Consumption: Android as an operating system consumes more power.
- Lost Tracks: With the coming up of new android versions, it becomes pretty challenging to adapt yourself to new Android SDK tools.
- Restrictions: Sometimes, there are restrictions on developers to follow the new standards for their existing applications. Failing to do so makes your existing app depreciated.

Challenges of Android Development

- Security: While developing an app, you should ensure that both user and system data are secure on the user's device.
- Compatibility: You should develop an app that can work on most Android versions. Not all people have the latest versions of Android running on their devices; hence making a supportive app would help you target a more significant user section.
- Performance: You should always try to develop an app that is responsive and doesn't lag. Many users don't have high-performance devices with greater RAM or memory space. So you should make sure that your app doesn't occupy much memory and doesn't create many junks.
- Screen Variants: Your app should be adaptive enough to adapt to any screen size or resolution. Some people may run your app on their tablets or their wide-screen smartphones. So your app should fit into those screen variants.
- Comments: You should always try to comment on your codes and work so that if anyone wants to add any feature in your application in the future, then they can easily understand what changes they need to make and where.
- Costly: As the development and testing consume more time, the cost of the application may increase, depending on the application's complexity and features.

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.

Programming Languages

Programming Languages used in Developing Android Applications

1. **Java**
2. **Kotlin**

Developing the Android Application using Kotlin is preferred by Google, as Kotlin is made an official language for Android Development, which is developed and maintained by JetBrains.

Previously before Java is considered the official language for Android Development. Kotlin is made official for Android Development in Google I/O 2017.

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.

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.

Android - Application Components

Activity:

- Activity life cycle
- Handle Activity State Changes
- Understand Tasks and Back Stack
- Processes and Application Lifecycle

Services:

- Types of Android Services
- The Life Cycle of Android Services

Content Provider:

- Content URI
- Operations in Content Provider
- Working of the Content Provider
- Creating a Content Provider

- **Broadcast Receiver:**
  - Implicit Broadcast Exceptions

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 {

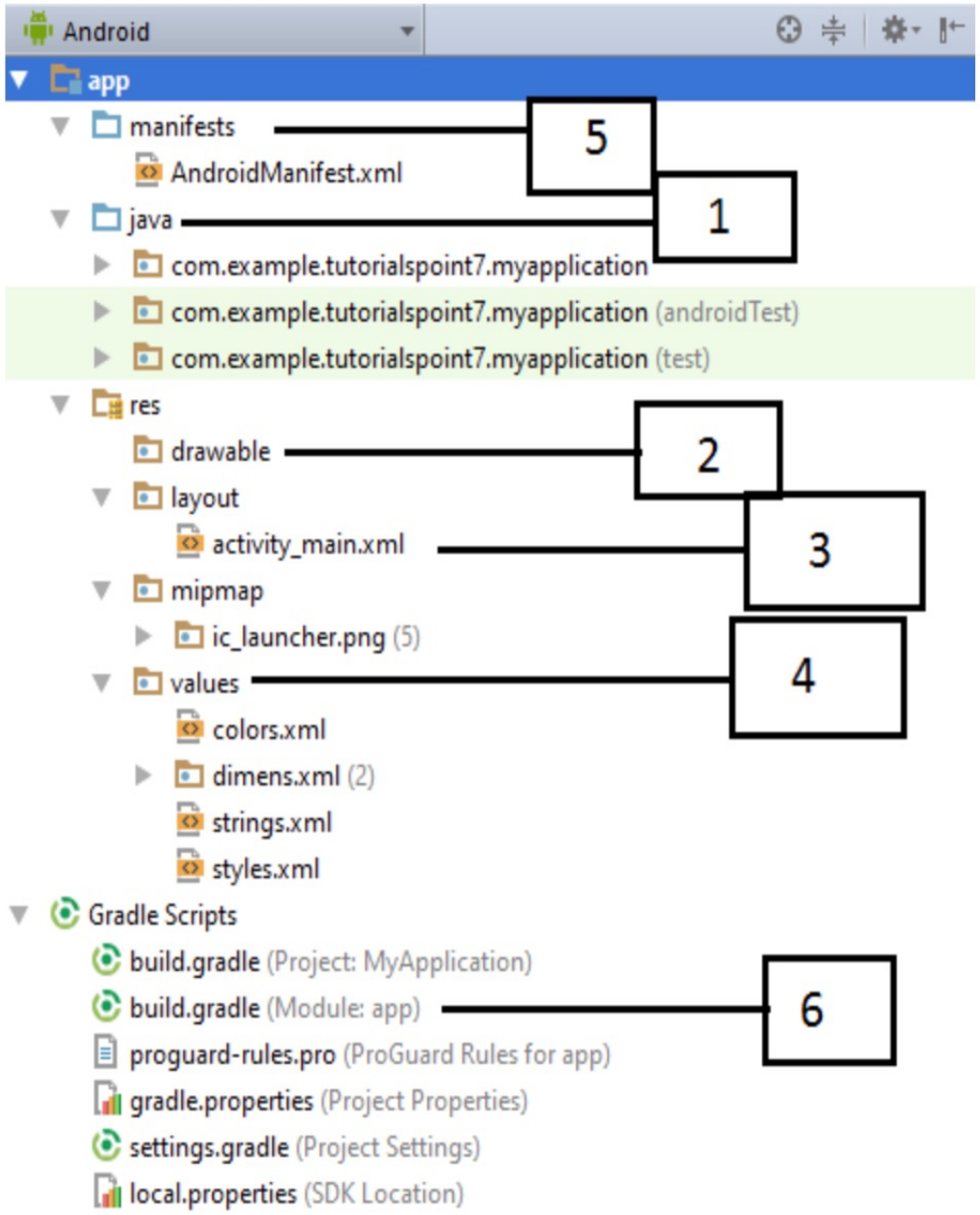
public void onCreate(){} }

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.



Sr.No.	Folder, File & Description
1	**Java** 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	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
3	**res/layout** This is a directory for files that define your app's user interface.
4	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	**Build.gradle** This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

Intents

It is a powerful inter-application message-passing framework.

They are extensively used throughout Android.

Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

Widgets

These are the small visual application components that you can find on the home screen of the devices.

They are a special variation of Broadcast Receivers that allow us to create dynamic, interactive application components for users to embed on their Home Screen.

Notifications are the application alerts that are used to draw the user's attention to some particular app event without stealing focus or interrupting the current activity of the user.

They are generally used to grab user's attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver.

Examples: E-mail popups, Messenger popups, etc.

Notifications

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.

Here is the default code generated by the application wizard for *Hello World!* application –

```
package com.example.helloworld;

Import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); } }
```

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 figured 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.

list of tags in Manifest file

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 Manifest File- Example

```
<?xml version="1.0" encoding="utf-8"?>

 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.tutorialspoint7.myapplication">

<application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
android:label="@string/app_name" android:supportsRtl="true"
android:theme="@style/AppTheme"> <activity android:name=".MainActivity">

<intent-filter> <action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter> </activity> </application> </manifest>
```

The Manifest File-Example

- 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.

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.

```
<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.

```
<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>
```