

Android

WackyCodes



August 30, 2019

WackyCodes by Shailendra

@Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_1*);

//---------- Nevigation Code and Toggle Button...  
  
 Toolbar toolbar = findViewById( R.id.*x\_ToolBar* );  
 setSupportActionBar( toolbar );  
  
 NavigationView navigationView = findViewById( R.id.*x\_Nav\_View* );  
 navigationView.setNavigationItemSelectedListener( this );

// Search Method setNavigationItemSelectedListener()...  
  
 w\_drawerLayout = findViewById( R.id.*x\_DrawerLayout* );  
 ActionBarDrawerToggle toggle =

new ActionBarDrawerToggle( this,w\_drawerLayout,toolbar,

R.string.*navigation\_Drawer\_Open*,R.string.*navigation\_Drawer\_close*);  
 w\_drawerLayout.addDrawerListener( toggle );  
 toggle.syncState();

}

**Context –**

1. **Introduction of Android** 
   1. What is Android?
   2. What is Open Handset Alliance (OHA)
   3. Features of Android
   4. Categories of Android applications
   5. History of Android
   6. Android Versions, Codename and API
   7. Android Architecture
   8. Android Core Building Blocks
2. **WackyCodes**

**Introduction of Android**

**Android** is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc.

It contains a **Linux-based Operating System**, **middleware** and **key mobile applications**.

**According to Wikipedia:-**

***Android is a mobile operating system developed by Google***. It is based on a modified version of the Linux kernel and other open source software, and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has developed Android TV for televisions, Android Auto for cars, and Wear OS for wearable, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

**1.1 What is Android?**

*Android is a software package and Linux based operating system for mobile devices such as tablet computers and smartphones.*



It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc which is covered in next page.

**1.2 What is Open Handset Alliance (OHA)**

It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.

It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Plateform.

**1.3 Features of Android**

After learning what is android, let's see the features of android. The important features of android are given below:

1) It is open-source.

2) Anyone can customize the Android Platform.

3) There are a lot of mobile applications that can be chosen by the consumer.

4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

*( This text is copied from* [*www.javatpoint.com*](http://www.javatpoint.com) *)*

**1.4 Categories of Android applications**

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

* Entertainment
* Tools
* Communication
* Productivity
* Personalization
* Music and Audio
* Social
* Media and Video
* Travel and Local etc.

**1.5 History of Android**

The history and versions of android are interesting to know. The code names of android ranges from A to P currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**, **Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **Kit Kat**, **Lollipop, Marshmallow, Nougat, Oreo** and **Pie**. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by co-workers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

**1.6 Android Versions, Codename and API**

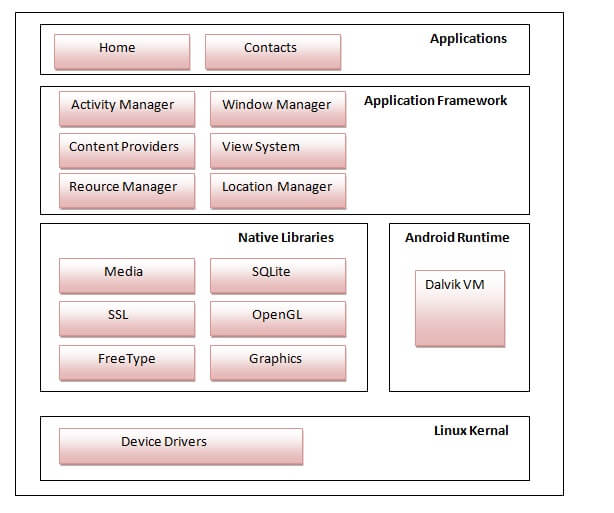
|  |  |  |
| --- | --- | --- |
| **Version** | **Code name** | **API Level** |
| 1.5 | Cupcake | 3 |
| 1.6 | Donut | 4 |
| 2.1 | Eclair | 7 |
| 2.2 | Froyo | 8 |
| 2.3 | Gingerbread | 9 and 10 |
| 3.1 and 3.3 | Honeycomb | 12 and 13 |
| 4.0 | Ice Cream Sandwitch | 15 |
| 4.1, 4.2 and 4.3 | Jelly Bean | 16, 17 and 18 |
| 4.4 | KitKat | 19 |
| 5.0 | Lollipop | 21 |
| 6.0 | Marshmallow | 23 |
| 7.0 | Nougat | 24-25 |
| 8.0 | Oreo | 26-27 |
| 9.0 | Pie | 28 |

**1.7 Android Architecture**

**android architecture** or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.



## 1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

## 2) Native Libraries

On the top of linux kernel, their are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

## 3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

## 4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

## 5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using Linux kernel.

**1.8 Android Core Building Blocks**



An android **component** is simply a piece of code that has a well-defined life cycle e.g. Activity, Receiver and Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

#### Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

#### View

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

#### Intent

Intent is used to invoke components. It is mainly used to:

* Start the service
* Launch an activity
* Display a web page
* Display a list of contacts
* Broadcast a message
* Dial a phone call etc.

For example, you may write the following code to view the webpage.

Intent intent=**new** Intent(Intent.ACTION\_VIEW);

intent.setData(Uri.parse("http://www.javatpoint.com"));

startActivity(intent);

#### Service

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

#### Content Provider

Content Providers are used to share data between the applications.

#### Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

#### AndroidManifest.xml

It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

#### Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

**Content:-**

1. **Create A Project –**

**Step 1: Create an Activity**

**Step 2: Connect XML file or Layout with Activity:**

**Step 3: Add reference of activity in AndroidManifest.xml file:**

1. **Add Fragment in our Activity –**
   1. **Add Fragment Layout in Main XML File**
   2. **Set Layout in Activity**
2. **Some Sort Technique to make our code Easy…**
   1. **w\_ShowToast( )**
   2. **w\_CheckInternetConnectivity( )**
      1. **w\_isInternetConnect( )**
   3. **w\_ShowAlertMessage( )**
      1. **w\_SetAlertForSingleButton( )**
      2. **w\_SetAlertForDoubleButton( )**
      3. **w\_SetAlertForTrippleButton( )**
   4. **w\_Validation( )**
      1. **w\_isEmailValid( )**
      2. **w\_isPassValid( )**
      3. **w\_isInformationValid( )**
   5. **Different between variable and method Naming.**
   6. **Define variables and given Name**
   7. **Create a XML File in easy but separate manner**
   8. **Create Separate class for separate big task.**
3. **Firebase**
   1. **How to connect Firebase**
   2. **Method for firebase connection**
   3. **w\_FirebaseSignIn( )**
   4. **w\_FirebaseSignUp( )**
4. **Kjdsnk**
5. **Jnkaksbk**
6. **Kasjdn**
7. **Kasjn**
8. **Create A Project -**

**Step 1: Create an Activity**

Extends *AppCompatActivity* after naming our class.

public class WackyTestActivity extends AppCompatActivity { // class name WackyTestActivity  
  
 private FrameLayout frameLayout;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate( savedInstanceState );  
 setContentView( R.layout.*activity\_wacky\_test* ); // xml Layout reference  
   
 }  
  
}

**Step 2: Connect XML file or Layout with Activity:**

File name : activity\_wacky\_test

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"

tools:context=".WackyTestActivity ">  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**Step 3: Add reference of activity in AndroidManifest.xml file:**

To launch our activity and connect each activity with others, we need to add reference of our activity in manifest file.

<application>

<activity  
 android:name=".WackyTestActivity ">  
  
 <intent-filter android:autoVerify="true">  
 <action android:name="android.intent.action.MAIN " />

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

</application>

**Done..!**

1. **Add Fragment in our Activity –**
   1. **Add Fragment Layout in Main XML File.**

<FrameLayout  
 android:id="@+id/framelayout\_activity"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp">  
  
</FrameLayout>

* 1. **Set Layout in Activity**

Private FrameLayout frameLayout;

// get reference in onCreate method.

frameLayout = findViewById(R.layout.framelayout\_activity);

// Create a Method of Fragment.

private void setFragment(Fragment fragment) {  
 FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();  
 fragmentTransaction.replace( frameLayout.getId(), fragment );  
 fragmentTransaction.commit();  
}

* 1. **:** Create a Fragment and call the setFragment() method in our onCreate() method inside our Activity and also pass the reference of Fragment class.

*For More detail about Fragment Check* [*www.javatpint.com*](http://www.javatpint.com) *or Check on Android official docs area.*

1. **Some Sort Technique to make our code Easy…**
   1. **w\_ShowToast( )**

// Toast message show method...  
private void w\_showToast(String s){  
 Toast.*makeText*(getActivity(), s, Toast.*LENGTH\_SHORT*).show();

// or  
 Toast.*makeText*(getActivity(), s, Toast.*LENGTH\_LONG*).show();

}

* 1. **w\_CheckInternetConnectivity( )**

First we need to create a new class to check whether is internet connected or not.

public class W\_CheckInternetConnection {  
  
  
 public boolean isConnected(Context context) {  
  
 ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo netinfo;  
 netinfo = cm.getActiveNetworkInfo();  
 if (netinfo != null && netinfo.isConnectedOrConnecting()) {  
 android.net.NetworkInfo wifi = cm.getNetworkInfo( ConnectivityManager.*TYPE\_WIFI*);  
 android.net.NetworkInfo mobile = cm.getNetworkInfo(ConnectivityManager.*TYPE\_MOBILE*);  
 if((mobile != null && mobile.isConnectedOrConnecting()) || (wifi != null && wifi.isConnectedOrConnecting()))  
 return true; else return false;  
 } else  
 return false;  
 }  
 public AlertDialog.Builder buildDialog(Context c) {  
  
 final AlertDialog.Builder builder = new AlertDialog.Builder(c);  
 builder.setTitle("No Internet Connection !");  
 builder.setMessage("Check Your Internet Connection.\nYou need to have Mobile Data or wifi...");  
  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.dismiss();  
 }  
  
 });  
  
 return builder;  
 }  
  
 public boolean checkInternet(Context context){  
 if(!isConnected(context)){  
 buildDialog(context).show();  
 return true;  
 }  
 else  
 return false;  
  
 }  
}

* + 1. **w\_isInternetConnect( )**

This is calling process from any activity or fragment.

private boolean w\_isInternetConnect() {  
 IJ\_CheckInternetConnection checkInternetCON = new IJ\_CheckInternetConnection();  
 if (checkInternetCON.checkInternet( getActivity() )) {  
 return false;  
 } else {  
 return true;  
 }  
  
}

* 1. **w\_ShowAlertMessage( )**

To show alert message we will create a new class and make new methods for different alert messages.

public class W\_AlertMessage\_Class {  
 // Method for showing alert Message...  
 public void w\_SetAlertForSingleButton (String alertMessage, String buttonText, Context context) {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder( context );  
 builder.setMessage( alertMessage );  
 builder.setCancelable( true );  
  
 builder.setPositiveButton( buttonText, new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 } );  
  
 AlertDialog alertDialog = builder.create();  
 alertDialog.show();  
 }  
  
 public void w\_SetAlertForDoubleButton (String alertMessage, String buttonText1, String buttonText2, Context context) {  
 final boolean returnMSG;  
 AlertDialog.Builder builder = new AlertDialog.Builder( context );  
 builder.setMessage( alertMessage );  
 builder.setCancelable( true );  
  
 builder.setPositiveButton( buttonText1, new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 } );  
  
 builder.setNegativeButton( buttonText2, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 dialogInterface.cancel();  
 }  
 } );  
  
 AlertDialog alertDialog = builder.create();  
 alertDialog.show();  
 }  
}

* + 1. **w\_SetAlertForSingleButton( )**
    2. **w\_SetAlertForDoubleButton( )**
    3. **w\_SetAlertForTrippleButton( )**
  1. **w\_Validation( )**

This method inside WackyMethods class, to check all validation.

* + 1. **w\_isEmailValid( )**

private boolean w\_isEmailValid(String wEmail, EditText wReference){  
 String emailRegex =  
 "^[a-zA-Z0-9\_+&\*-]+(?:\\."+  
 "[a-zA-Z0-9\_+&\*-]+)\*@" +  
 "(?:[a-zA-Z0-9-]+\\.)+[a-z" +  
 "A-Z]{2,7}$";  
 Pattern pat = Pattern.*compile*(emailRegex);  
 boolean bool = pat.matcher(wEmail).matches();  
  
 if (TextUtils.*isEmpty*( wEmail )) {  
 wReference.setError( "Please Enter Email! " );  
 return false;  
 } else if (!bool){  
 wReference.setError( "Please Enter Valid Email! " );  
 return false;  
 }  
 return true;  
}

* + 1. **w\_isPassValid( )**
    2. **w\_isInformationValid( )**
  1. **Different between variable and method Naming.**
  2. **Define variables and given Name**
  3. **Create a XML File in easy but separate manner**
  4. **Create Separate class for separate big task.**

1. **Firebase**
   1. **How to connect Firebase**
   2. **Method for firebase connection**
   3. **w\_FirebaseSignIn( )**
   4. **w\_FirebaseSignUp( )**