**Android Operating System 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 there is the Linux Kernel. Android was created on the open source kernel of Linux. 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. The features of Linux kernel are:

## *Security:* The Linux kernel handles the security between the application and the system.

## *Memory Management:* It efficiently handles the memory management thereby providing the freedom to develop apps.

## *Process Management:* It manages the process well, allocates resources to processes whenever they need them.

## *Network Stack:* It effectively handles the network communication.

## *Driver Model:* It ensures that the application works. Hardware manufacturers can build their drivers into Linux build.

***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. A summary of some key core Android libraries available to the Android developer is as follows:

## *Open GL(graphics library):* This cross-language, cross-platform application program interface (API) is used to produce 2D and 3D computer graphics.

## *WebKit:* This open source web browser engine provides all the functionality to display web content and to simplify page loading.

## *Media frameworks:* These libraries allow you to play and record audio and video.

## *Secure Socket Layer (SSL):* These libraries are there for Internet security.

## *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 Delvik VM uses the device’s underlying Linux kernel to handle low-level functionality,including security,threading and memory management. 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**

Android applications can be found at the topmost layer. At application layer we write our application to be installed on this layer only. Examples of applications are Games, Messages, Contacts etc.

# Android Application lifecycles:

***The Out of Memory:*** To manage limited system resources the Android system can terminate running applications. Each application is started in a new process with a unique ID under a unique user. If the Android system needs to free up resources it follows a simple set of rules. If the Android system needs to terminate processes it follows the following priority system.

| **Process status** | **Description** | **Priority** |
| --- | --- | --- |
| Foreground | An application in which the user is interacting with an activity, or which has an service which is bound to such an activity. Also if a service is executing one of its lifecycle methods or a broadcast receiver which runs its onReceive() method. | 1 |
| Visible | User is not interacting with the activity, but the activity is still (partially) visible or the application has a service which is used by a inactive but visible activity. | 2 |
| Service | Application with a running service which does not qualify for 1 or 2. | 3 |
| Background | Application with only stopped activities and without a service or executing receiver. Android keeps them in a least recent used (LRU) list and if requires terminates the one which was least used. | 4 |
| Empty | Application without any active components. | 5 |

All processes in the empty list are added to a *least recently used* list (LRU list). The processes which are at the beginning of this lists will be the ones killed by the out-of-memory killer. If an application is restarted by the user, its gets moved to the end of this queue. If it reaches the lowest prio again.

### [*Application*](https://www.vogella.com/tutorials/AndroidLifeCycle/article.html#application): You can specify a custom application class in your Android manifest file.The application object is already the first components started. It is also always the last component of the application, which is terminated.This object provides the following main life-cycle methods:

*onCreate()* - called before the first components of the application starts

*onLowMemory()* - called when the Android system requests that the application cleans up memory

*onTrimMemory()* - called when the Android system requests that the application cleans up memory. This message includes an indicator in which position the application is. For example the constant TRIM\_MEMORY\_MODERATE indicates that the process is around the middle of the background LRU list; freeing memory can help the system keep other processes running later in the list for better overall performance.

*onTerminate()* - only for testing, not called in production

*onConfigurationChanged()* - called whenever the configuration changes

***Android Activity lifecycles:***

It is very important to have a basic understanding of the lifecycle for an activity in Android in order to efficiently manage resources on limited devices, and to ensure a seamless response for the user.A general overview of the lifecycle of an application is shown in the following figure



|  |
| --- |
|  |

# The seven methods contained in square boxes in the preceding diagram are called the lifecyle methods because they govern the lifecycles of Android applications. Their properties are summarized below-

|  |
| --- |
| Activity Lifecycle Methods |
| Method | Description |
| [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) | Called when activity first created |
| [onRestart()](http://developer.android.com/reference/android/app/Activity.html#onRestart()) | Called after activity stopped, prior to restarting |
| [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()) | Called when activity is becoming visible to user |
| [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()) | Called when activity starts interacting with user |
| [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) | Called when a previous activity is about to resume |
| [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop()) | Called when activity no longer visible to user |
| [onDestroy()](http://developer.android.com/reference/android/app/Activity.html#onDestroy()) | Final call received before activity is destroyed |

**Android Programming for a calculator:**

***import android.support.v7.app.AppCompatActivity;***

***import android.os.Bundle;***

***import android.view.View;***

***import android.widget.EditText;***

***import android.widget.TextView;***

***public class MainActivity extends AppCompatActivity {***

 ***EditText t1,t2;***

 ***TextView tv;***

 ***@Override***

 ***protected void onCreate(Bundle savedInstanceState) {***

 ***super.onCreate(savedInstanceState);***

 ***setContentView(R.layout.activity\_main);***

 ***t1=(EditText)findViewById(R.id.txt1);***

 ***t2=(EditText)findViewById(R.id.txt2);***

 ***tv=(TextView) findViewById(R.id.tv);***

 ***}***

 ***public void doAdd(View v)***

 ***{***

 ***try***

 ***{***

 ***int a=Integer.parseInt("" + t1.getText());***

 ***int b=Integer.parseInt("" + t2.getText());***

 ***int sum=a+b;***

 ***tv.setText("" + sum);***

 ***}***

 ***catch (Exception ex)***

 ***{***

 ***System.out.println(ex);***

 ***tv.setText(ex.getMessage());***

 ***}***

 ***}***

 ***public void doSub(View v)***

 ***{***

 ***try***

 ***{***

 ***int a=Integer.parseInt("" + t1.getText());***

 ***int b=Integer.parseInt("" + t2.getText());***

 ***int sub=a-b;***

 ***tv.setText("" + sub);***

 ***}***

 ***catch (Exception ex)***

 ***{***

 ***System.out.println(ex);***

 ***tv.setText(ex.getMessage());***

 ***}***

 ***}***

 ***public void doMulty(View v)***

 ***{***

 ***try***

 ***{***

 ***int a=Integer.parseInt("" + t1.getText());***

 ***int b=Integer.parseInt("" + t2.getText());***

 ***int mul=a\*b;***

 ***tv.setText("" + mul);***

 ***}***

 ***catch (Exception ex)***

 ***{***

 ***System.out.println(ex);***

 ***tv.setText(ex.getMessage());***

 ***}***

 ***}***

 ***public void doDivide(View v)***

 ***{***

 ***try***

 ***{***

 ***int a=Integer.parseInt("" + t1.getText());***

 ***int b=Integer.parseInt("" + t2.getText());***

 ***int div=a/b;***

 ***tv.setText("" + div);***

 ***}***

 ***catch (Exception ex)***

 ***{***

 ***System.out.println(ex);***

 ***tv.setText(ex.getMessage());***

 ***}***

 ***}***

***}***

**Android programming to read and play audio file:**

import android.media.MediaPlayer;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.widget.MediaController;

import android.widget.VideoView;

public class MainActivity extends Activity {

 @Override

 protected void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);

 setContentView(R.layout.activity\_main);

 MediaPlayer mp=new MediaPlayer();

 try{

 mp.setDataSource("/sdcard/Music/maine.mp3"); //Write your location here

 mp.prepare();

 mp.start();

 }catch(Exception e){e.printStackTrace();}

 }

 @Override

 public boolean onCreateOptionsMenu(Menu menu) {

 // Inflate the menu; this adds items to the action bar if it is present.

 getMenuInflater().inflate(R.menu.activity\_main, menu);

 return true;

 }

 }

**Android Menu system:**

There are 3 types of menus in Android:

1. Option Menu
2. Context Menu
3. Pop-up Menu

**Option Menu**



The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a overall impact on the app, such as Search, Compose Email and Settings.

**Context Menu**



A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

**PopUp Menu**



A popup menu displays a list of items in a vertical list that is anchored(sticked) to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command.

**Android APIs:**

API is an acronym for “application program interface.” It’s a technical development environment that enables access to another party’s application or platform.

The most famous, and most often used by mobile developers, is Facebook’s API. It allows mobile developers limited access to the profile identity of Facebook members to verify identification on login. It enables Facebook members to sign up for a third-party app like Candy Crush using their Facebook account.

Like Facebook, Twitter has an API that can also be used by third-party mobile developers to identify their users on sign up and to enable users to post content easily from an app to their Twitter account.

Google Maps provides a hugely popular API that enables mapping and location services in third-party applications. Here are a few other popular APIs to know about:

* YouTube: provides access to their rich repository of video content
* AccuWeather: offers access to weather information, which is very popular on mobile
* Flickr: enables access to a large library of photography

In short, APIs enable apps to interact with each other. They can enhance the [user experience](https://www.upwork.com/hire/ux-designers/) and might even help fuel the popularity of your app.

*Android Media APIs:*

The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw resources), from standalone files in the filesystem, or from a data stream arriving over a network connection, all using [MediaPlayer](https://developer.android.com/reference/android/media/MediaPlayer.html) APIs.

**Design a User registration form with 3 text field and one submit button and insert values:**

**xml layout Coding(for designing the interface):**

 <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"

 android:paddingBottom="@dimen/activity\_vertical\_margin"

 android:paddingLeft="@dimen/activity\_horizontal\_margin"

 android:paddingRight="@dimen/activity\_horizontal\_margin"

 android:paddingTop="@dimen/activity\_vertical\_margin"

 tools:context="com.userregistrationformphpmysql\_android\_examples.com.MainActivity"

 android:background="#02c3d4" >

 <TextView

 android:id="@+id/textView1"

 android:layout\_width="fill\_parent"

 android:layout\_height="wrap\_content"

 android:layout\_alignParentTop="true"

 android:layout\_centerHorizontal="true"

 android:text="REGISTER HERE"

 android:gravity="center"

 android:textAppearance="?android:attr/textAppearanceLarge"

 android:textColor="#fbfefd"

 />

 <EditText

 android:id="@+id/editText1"

 android:layout\_width="fill\_parent"

 android:layout\_height="50dp"

 android:layout\_below="@+id/textView1"

 android:layout\_centerHorizontal="true"

 android:layout\_marginTop="24dp"

 android:ems="10"

 android:hint="ENTER YOUR NAME"

 android:background="#fbfefd"

 android:gravity="center" >

 </EditText>

 <EditText

 android:id="@+id/editText2"

 android:layout\_width="fill\_parent"

 android:layout\_height="50dp"

 android:layout\_below="@+id/editText1"

 android:layout\_centerHorizontal="true"

 android:layout\_marginTop="24dp"

 android:ems="10"

 android:hint="ENTER YOUR EMAIL ID"

 android:inputType="textEmailAddress"

 android:background="#fbfefd"

 android:gravity="center" >

 <requestFocus />

 </EditText>

 <EditText

 android:id="@+id/editText3"

 android:layout\_width="fill\_parent"

 android:layout\_height="50dp"

 android:layout\_marginTop="24dp"

 android:layout\_below="@+id/editText2"

 android:layout\_centerHorizontal="true"

 android:ems="10"

 android:hint="ENTER YOUR PASSWORD"

 android:inputType="textPassword"

 android:background="#fbfefd"

 android:gravity="center" />

 <Button

 android:id="@+id/button1"

 android:layout\_width="fill\_parent"

 android:layout\_height="wrap\_content"

 android:layout\_below="@+id/editText3"

 android:layout\_centerHorizontal="true"

 android:layout\_marginTop="24dp"

 android:text="REGISTER"

 android:background="#d8dbdb"

 />

</RelativeLayout>

**Code for MainActivity.java file(for reading and storing user data)**

package com.userregistrationformphpmysql\_android\_examples.com;

import java.io.BufferedInputStream;

import java.io.IOException;

import java.io.InputStream;

import android.app.Activity;

import android.os.AsyncTask;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

public class MainActivity extends Activity {

 EditText name,email,password ;

 String Getname,GetEmail,GetPassword ;

 Button register ;

 String DataParseUrl = "http://androidblog.esy.es/test/insert-registration-data.php" ;

 Boolean CheckEditText ;

 String Response;

 HttpResponse response ;

 @Override

 protected void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);

 setContentView(R.layout.activity\_main);

 name = (EditText)findViewById(R.id.editText1);

 email = (EditText)findViewById(R.id.editText2);

 password = (EditText)findViewById(R.id.editText3);

 register = (Button)findViewById(R.id.button1) ;

 register.setOnClickListener(new View.OnClickListener() {

 @Override

 public void onClick(View v) {

 // TODO Auto-generated method stub

 GetCheckEditTextIsEmptyOrNot();

 if(CheckEditText){

 SendDataToServer(Getname, GetEmail, GetPassword);

 }

 else {

 Toast.makeText(MainActivity.this, "Please fill all form fields.", Toast.LENGTH\_LONG).show();

 }

 }

 });

 }

 public void GetCheckEditTextIsEmptyOrNot(){

 Getname = name.getText().toString();

 GetEmail = email.getText().toString();

 GetPassword = password.getText().toString();

 if(TextUtils.isEmpty(Getname) || TextUtils.isEmpty(GetEmail) || TextUtils.isEmpty(GetPassword)) {

 CheckEditText = false;

 }

 else {

 CheckEditText = true ;

 }

 }

 public void SendDataToServer(final String name, final String email, final String password){

 class SendPostReqAsyncTask extends AsyncTask<String, Void, String> {

 @Override

 protected String doInBackground(String... params) {

 String QuickName = name ;

 String QuickEmail = email ;

 String QuickPassword = password;

 List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();

 nameValuePairs.add(new BasicNameValuePair("name", QuickName));

 nameValuePairs.add(new BasicNameValuePair("email", QuickEmail));

 nameValuePairs.add(new BasicNameValuePair("password", QuickPassword));

 try {

 HttpClient httpClient = new DefaultHttpClient();

 HttpPost httpPost = new HttpPost(DataParseUrl);

 httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

 HttpResponse response = httpClient.execute(httpPost);

 HttpEntity entity = response.getEntity();

 } catch (ClientProtocolException e) {

 } catch (IOException e) {

 }

 return "Data Submit Successfully";

 }

 @Override

 protected void onPostExecute(String result) {

 super.onPostExecute(result);

 Toast.makeText(MainActivity.this, "Data Submit Successfully", Toast.LENGTH\_LONG).show();

 }

 }

 SendPostReqAsyncTask sendPostReqAsyncTask = new SendPostReqAsyncTask();

 sendPostReqAsyncTask.execute(name, email, password);

 }

 }

**Code for insert-registration-data.php file.(for database operation)**

<?php

//This script is designed by Android-Examples.com

//Define your host here.

$hostname = "mysql.hostinger.in";

//Define your database username here.

$username = "u288012116\_new";

//Define your database password here.

$password = "1234567890";

//Define your database name here.

$dbname = "u288012116\_new";

 $con = mysqli\_connect($hostname,$username,$password,$dbname);

 $name = $\_POST['name'];

 $email = $\_POST['email'];

 $password = $\_POST['password'];

 $Sql\_Query = "insert into registration (name,email,password) values ('$name','$email','$password')";

 if(mysqli\_query($con,$Sql\_Query)){

 echo 'Data Inserted Successfully';

 }

 else{

 echo 'Try Again';

 }

 mysqli\_close($con);

?>

Output:



# Login form coding:-(go to the link): 🡪 <https://www.tutorialspoint.com/android/android_login_screen.htm>

# 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. |