**1 Eclipse IDE For Android App Development?**

Android Application Development can be done using Android Studio as well as Eclipse IDE. We can create android applications in Eclipse IDE using the ADT plugin. Eclipse is preferred for creating small android applications. Eclipse IDE is an open-source software used by developers, it contains a variety of plugins to develop software in different programming languages. We will be using Eclipse IDE to set up Android App Development. First, we need to install Eclipse IDE, and then we will be setting it up for Android App Development.

**Steps to Install Eclipse IDE**

To install Eclipse IDE, click on [Download Eclipse](https://www.eclipse.org/downloads/)

Download [JDK (Java Development Kit)](https://www.oracle.com/java/technologies/downloads/) and [Android Studio](https://developer.android.com/studio) as well.

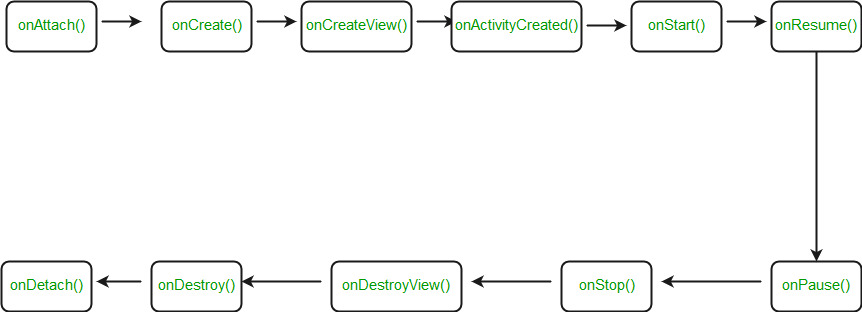
In File Explorer, go to Downloads – “Eclipse IDE” will be downloaded.

Open Eclipse IDE, choose Eclipse IDE for Java Developers and Install.

Eclipse IDE environment is ready, now it’s time to set up Android Development.

**Steps to Setup Eclipse IDE for Android App Development**

And we use some steps for this setup development

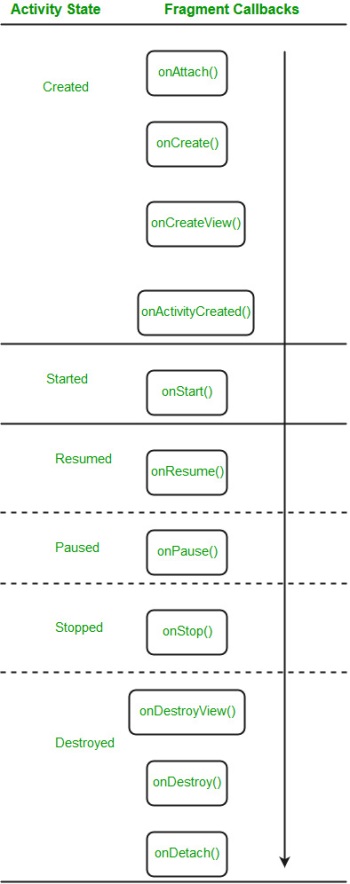


**Fragment**

A Fragment is a piece of an activity which enable more modular activity design. A fragment encapsulates functionality so that it is easier to reuse within activities and layouts.  
Android device exists in a variety of screen sizes and densities. Fragments simplify the reuse of components in different layouts and their logic. You can build single-pane layouts for handsets (phones) and multi-pane layouts for tablets. You can also use fragments also to support different layout for landscape and portrait orientation on a smartphone.

The below image shows how two UI modules defined by fragments can be combined into one activity for a tablet design but separated for a handset design

**Handling the Fragment Lifecycle**



**2 Intents**

In Android, it is quite usual for users to witness a jump from one application to another as a part of the whole process, for example, searching for a location on the browser and witnessing a direct jump into Google Maps or receiving payment links in Messages Application (SMS) and on clicking jumping to PayPal or GPay (Google Pay). This process of taking users from one application to another is achieved by passing the Intent to the system. Intents, in general, are used for navigating among various activities within the same application, but note, is not limited to one single application, i.e., they can be utilized from moving from one application to another as well.

Intents could be Implicit, for instance, calling intended actions, and explicit as well, such as opening another activity after some operations like onClick or anything else. Below are some applications of Intents:

**There are two types of intents in android**

* Implicit
* Explicit

**Implicit Intent**

[Implicit Intent](https://www.geeksforgeeks.org/android-implicit-and-explicit-intents-with-examples/) doesn’t specify the component. In such a case, intent provides information on available components provided by the system that is to be invoked. For example, you may write the following code to view the webpage.

**Explicit Intent**

[Explicit Intent](https://www.geeksforgeeks.org/android-implicit-and-explicit-intents-with-examples/) specifies the component. In such a case, intent provides the external class to be invoked.

**3 Notification**

is a kind of message, alert, or status of an application (probably running in the background) that is visible or available in the Android’s UI elements. This application could be running in the background but not in use by the user. The purpose of a notification is to notify the user about a process that was initiated in the application either by the user or the system. This article could help someone who’s trying hard to create a notification for developmental purposes.

Notifications could be of various formats and designs depending upon the developer. In General, one must have witnessed

**These four types of notifications**:

* Status Bar Notification (appears in the same layout as the current time, battery percentage)
* Notification drawer Notification (appears in the drop-down menu)
* Heads-Up Notification (appears on the overlay screen, ex: Whatsapp notification, OTP messages)
* Lock-Screen Notification (I guess you know it)

**Components of and android screen application**

There are some necessary building blocks that an Android application consists of. These loosely coupled components are bound by the application manifest file which contains the description of each component and how they interact.

The manifest file also contains the app’s metadata, its hardware configuration, and platform requirements, external libraries, and required permissions.

**There are the following main components of an android app.**

1. **Activities**

Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user’s actions. An activity is implemented as a subclass of class Activity.

**2. Services**

Services are like invisible workers of our app. These, updating your data sources and Activities, triggering Notification, components run at the backend and also broadcast Intents. They also perform some tasks when applications are not active. A service can be used as a subclass of class Service:

3**. Content Providers**

It is used to manage and persist the application data also typically interacts with the SQL database. They are also responsible for sharing the data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured.

A content provider should be a sub-class of the class ContentProvider.

4. **Broadcast Receivers**

They are known to be intent listeners as they enable your application to listen to the Intents that satisfy the matching criteria specified by us. Broadcast Receivers make our application react to any received Intent thereby making them perfect for creating event-driven applications. Its responsible for broadcasting of received information from content provider.

5. **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.

6. **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.

7. **Notifications**

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.

**UI screen components**

A typical user interface of an android application consists of action bar and the application content area.

* Main Action Bar
* View Control
* Content Area
* Split Action Bar

These components have also been shown in the image below −

**Anroid UI**

Understanding Screen Components

The basic unit of android application is the activity. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

**View and ViewGroups**

An activity is consist of views. A view is just a widget that appears on the screen. It could be button e.t.c. One or more views can be grouped together into one GroupView. Example of ViewGroup includes layouts.

**Types of layout**

There are many types of layout. Some of which are listed below −

* Linear Layout
* Absolute Layout
* Table Layout
* Frame Layout
* Relative Layout

Linear layout is further divided into horizontal and vertical layout. It means it can arrange views in a single column or in a single row. Here is the code of linear layout(vertical) that includes a text view.

Kuch layout bhi yad k mtlb bhi malomhona chahiye

1 layout\_width

Specifies the width of the View or ViewGroup

2 layout\_height

Specifies the height of the View or ViewGroup

3 layout\_marginTop

Specifies extra space on the top side of the View or ViewGroup

4 layout\_marginBottom

Specifies extra space on the bottom side of the View or ViewGroup

5 layout\_marginLeft

Specifies extra space on the left side of the View or ViewGroup

6 layout\_marginRight

Specifies extra space on the right side of the View or ViewGroup

7 layout\_gravity

Specifies how child Views are positioned

8 layout\_weight

Specifies how much of the extra space in the layout should be allocated to the View

ooooooooooooooooooo

**3 Adapting to Display Orientation**

One of the key features of modern smartphones is their ability to switch screen orientation

Android supports two screen orientations: portrait and landscape

By default, when you change the display orientation of your Android device, the current activity that is displayed will automatically redraw its content in the new orientation.

This is because the onCreate() event of the activity is fired whenever there is a change in display orientation.

When you change the orientation of your Android device, your current activity is actually destroyed and then re-created.

**Adapting to Display Orientation example**

As you can observe in landscape mode, a lot of empty space on the right of the screen could be used.

Furthermore, any additional views at the bottom of the screen would be hidden when the screen orientation is set to landscape.

Handling changes in screen orientation

In general, you can employ two techniques to handle changes in screen orientation:

**Anchoring -** The easiest way is to "anchor" your views to the four edges of the screen. When the screen orientation changes, the views can anchor nearly to the edges.

Resizing and repositioning - Whereas anchoring and centralizing are simple techniques to ensure that views can handle changes in screen orientation, the ultimate technique is resizing each and every view according to the current screen orientation.

**Anchoring Views**

Anchoring could be easily achieved by using RelativeLayout

Consider the following

main.xml containing five Button views embedded within the <RelativeLayout> element:

When the screen orientation changes to landscape mode

the four buttons are aligned to the four edges of the screen, and

the center button is centered in the middle of the screen with its width fully stretched.

**Resizing and Repositioning**

Apart from anchoring your views to the four edges of the screen, an easier way to customize the UI based on screen orientation is to create a separate resize/layout folder containing the XML files for the Ul of each orientation.

To support landscape mode, you can create a new folder in the res folder and name it as layout-land (representing landscape).

**4 Managing Changes to Screen Orientation**

When configurations change during run time (such as screen orientation, keyboard availability, and language), Android usually destroys an application's existing Activity or Fragment and recreates it.

Android does this so that applications can reload its resources based on the new configuration. The restart behavior helps applications to adapt new configurations by automatically reloading the application with alternative resources that match the new device configuration.

Proper handling of orientation changes makes rich user experience (not lost UI state) for the application and it also avoids memory leaks.

**How to handle it?**

To handle these configuration changes, Android provides callbacks to save your application state before destroying either Activity or Fragment. In the same it also provides to restore the application state when it is recreating them.

There are different options to handle the orientation changes:

* Lock screen orientation
* Prevent Activity to recreated
* Save basic state
* Save complex objects

**Lock screen orientation**

To lock the screen orientation change of any screen (activity) of your android application makes your activity display only in one mode i.e. either Landscape or Portrait. This is the simplest way to handle screen orientation but not generally recommended.

For this you need to add the below line in your projects AndroidManifest.xml. Add the below line along with your activity entry in the AndroidManifest file.

**Prevent Activity to recreated**

Another most common solution to dealing with orientation changes by setting the android:configChanges flag on your Activity in AndroidManifest.xml. Using this attribute your Activities won’t be recreated and all your views and data will still be there after orientation change.

This attribute informs the Android system that you are going to handle orientation and screenSize changes for this Activity. So instead of destroying and recreating your Activity, Android will just rotate the screen and invoke one of the lifecycle callback methods which is onConfigurationChanged(Configuration).

**Save basic state**

This is the most common situation to save the basic data of your Activity or Fragment during orientation change. You can save Primitive data such as String, Boolean, Integers or Parcelable objects in a Bundle during the orientation change and read the same data when Activity recreated.

Saving and restoring the data works using two Activity lifecycle methods called onSaveInstanceState() and onRestoreInstanceState().

To save the state information override onSaveInstanceState() method and add key-value pairs to the Bundle object that is saved in the event that your activity is destroyed unexpectedly. This method gets called before onStop().

To recover your saved state from the Bundle override onRestoreInstanceState() method. This is called after onStart() and before onResume().

**Save complex objects**

Override onRetainNonConfigurationInstance()and getLastNonConfigurationInstance() Prior to Honeycomb’s release, the recommended means of transferring active objects across Activity instances was to override the onRetainNonConfigurationInstance() and getLastNonConfigurationInstance() methods. After API level 13 these methods have been deprecated in favor of the more Fragment’s setRetainInstance(boolean) capability, which provides a much cleaner and modular means of retaining objects during configuration changes.

**5 Utilizing the Action Bar**

Action Bar is one of the important parts of any application, whether it’s a web application or a mobile app. Today we will learn how to implement an action bar in android apps using the ActionBar component.

**ActionBar in Android** with Example

In Android applications, ActionBar is the element present at the top of the activity screen. It is a salient feature of a mobile application that has a consistent presence over all its activities. It provides a visual structure to the app and contains some of the frequently used elements for the users. Android ActionBar was launched by Google in 2013 with the release of Android 3.0(API 11). Before that, the name of this top most visual element was AppBar. AppBar contains only the name of the application or current activity. It was not very much useful for the users and developers also have negligible option to customize it.

Google announced a support library along with the introduction of ActionBar. This library is a part of AppCompat and its purpose is to provide backward compatibility for older versions of Android and to support tabbed interfaces. All applications that use the default theme provided by the Android(Theme.AppCompat.Light.DarkActionBar), contains an ActionBar by default. However, developers can customize it in several ways depending upon their needs. **Components included in the ActionBar are:**

1. App Icon: Display the branding logo/icon of the application.
2. View Controls: Section that displays the name of the application or current activity. Developers can also include spinner or tabbed navigation for switching between views.
3. Action Button: Contains some important actions/elements of the app that may be required to the users frequently.
4. Action Overflow: Include other actions that will be displayed as a menu.

**How to Create Interfaces in Android Studio?**

Interfaces are a collection of constants, methods(abstract, static, and default), and nested types. All the methods of the interface need to be defined in the class. The interface is like a Class. The interface keyword is used to declare an interface.

public interface AdapterCallBackListener {

void onRowClick(String searchText);

}

public interface OnFragmentInteractionListener {

void onFragmentInteraction();

}

So basically in android, there are two types of interfaces we can create and we use frequently.

* Creating Java Interface
* Creating Kotlin Interface

So in this article, we are going to create both Java and Kotlin Interface in Android studio.

**6 Creating Java Interface in Android Studio**

Like a class, a Interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

Interfaces specify what a class must do and not how. It is the blueprint of the class.

An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.

If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.

A Java library example is Comparator Interface. If a class implements this interface, then it can be used to sort a collection.

Syntax:

interface <interface\_name> {

// declare constant fields

// declare methods that abstract

// by default.

}

To declare an interface, use the interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static, and final by default. A class that implements an interface must implement all the methods declared in the interface. To implement interface use implements keyword.

**Step by Step Implementation**

Step 1: Go to Android Studio and open the project in Android mode as shown in the below image.

Step 2: Now go to the app > java > your package name > right-click > New > Java Class as shown in the below image.

Step 3: After completing step 2 a pop-up screen will arise like below. Here enter your interface name and choose the Interface and click the Enter button.

After completing the above steps successfully you can find your Java interface here. Go to the app > java > your package name > GeeksforGeeks.java. And you can write your own Java code here.

**8 Notifications in Android with Example**

Notification is a kind of message, alert, or status of an application (probably running in the background) that is visible or available in the Android’s UI elements. This application could be running in the background but not in use by the user. The purpose of a notification is to notify the user about a process that was initiated in the application either by the user or the system. This article could help someone who’s trying hard to create a notification for developmental purposes.

Notifications could be of various formats and designs depending upon the developer. In General, one must have witnessed these four types of notifications:

Status Bar Notification (appears in the same layout as the current time, battery percentage)

Notification drawer Notification (appears in the drop-down menu)

Heads-Up Notification (appears on the overlay screen, ex: Whatsapp notification, OTP messages)

Lock-Screen Notification (I guess you know it)

**8 What is Android View?**

A View is a simple building block of a user interface. It is a small rectangular box that can be TextView, EditText, or even a button. It occupies the area on the screen in a rectangular area and is responsible for drawing and event handling. View is a superclass of all the graphical user interface components.

**Why and How to use the View in Android?**

Now you might be thinking what is the use of a View. So, the use of a view is to draw content on the screen of the user’s Android device. A view can be easily implemented in an Application using the java code. Its creation is more easy in the XML layout file of the project. Like, the project for hello world that we had made initially.

If you have not tried it, refer DataFlair hello world app in Android.

**Types of Android Views**

Another thing that might now come to your mind must be, “what are the available types of view in Android that we can use?”

For that, we’ll see all these types one by one as follows:

* **TextView**
* **EditText**
* **Button**
* **Image Button**
* **Date Picker**
* **RadioButton**
* **CheckBox buttons**
* **Image View**

And there are some more components. Learn more about Android UI Controls.

**Another important feature in Android is ViewGroup which is as follows.**

**What is Android View Group?**

A View Group is a subclass of the ViewClass and can be considered as a superclass of Layouts. It provides an invisible container to hold the views or layouts. ViewGroup instances and views work together as a container for Layouts. To understand in simpler words it can be understood as a special view that can hold other views that are often known as a child view.

Following are certain commonly used subclasses for ViewGroup:

* LinearLayout
* RelativeLayout
* FrameLayout
* GridView
* ListView

**9 Persisting date**

Persisting data is an important topic in application development because users typically expect to reuse data in the future. For Android, there are primarily three basic ways of persisting data:

A lightweight mechanism known as shared preferences to save small chunks of data

**Traditional file systems**

A relational database management system through the support of SQLite databases

**Kinds of Data Persistence in Android**

In general, there you can store data in only two ways. Either you have to save it locally on your mobile device or keep it somewhere else, like on an SD card or cloud storage. So, we can say data persistence in Android is of two types. The first one is, On-device and the Second, Off device.

Besides that, functionality-wise, there are three main categories for data persistence and five categories by sub-categories. So you will see here five categories, and for today we will sort this article into 3 categories.

**Five Different Types of Persistent Data in an Android Application**

Here we will show you the different persistent data types of all methods available for storing Android application data.

1. Data Storage / File IO

2. Shared preference

3. SQLite Database

4. Custom Content Provider

5. Web Services / Cloud Storage

**10 Sharing data android**

Android uses the action ACTION\_SEND to send data from one activity to another, even across process boundaries. You need to specify the data and its type. The system automatically identifies the compatible activities that can receive the data and displays them to the user. In the case of the intent resolver, if only one activity can handle the intent, that activity immediately starts.

**How to Send Data From One Activity to Second Activity in Android?**

This article aims to tell and show how to “Send the data from one activity to second activity using Intent”. In this example, we have two activities, activity\_first which are the source activity, and activity\_second which is the destination activity. We can send the data using the putExtra() method from one activity and get the data from the second activity using the getStringExtra() method.

**11 Sending SMS Messages.**

Sending a Text Message Over the Phone Using SmsManager in Android

This article is about sending a text SMS over the phone using the SMSManager class in an Android application. For this, a basic knowledge of the fundamentals of android app development, creating a new project, running an android app, Views, and handling of click event buttons is required.

SMSManager class manages operations like sending a text message, data message, and multimedia messages (MMS). For sending a text message method sendTextMessage() is used likewise for multimedia message sendMultimediaMessage() and for data message sendDataMessage() method is used. The details of each function are:

**Function**

sendTextMessage() sendTextMessage(String destinationAddress,

String scAddress, String text, PendingIntent sentIntent,

PendingIntent deliveryIntent, long messageId)

sendDataMessage() sendDataMessage(String destinationAddress,

String scAddress, short destinationPort, byte[] data,

PendingIntent sentIntent, PendingIntent deliveryIntent)

sendMultimediaMessage() sendMultimediaMessage(Context context,

Uri contentUri, String locationUrl,

Bundle configOverrides, PendingIntent sentIntent.

**12 Sending Emails**

Email is messages distributed by electronic means from one system user to one or more recipients via a network.

Before starting Email Activity, You must know Email functionality with intent, Intent is carrying data from one component to another component with-in the application or outside the application.

To send an email from your application, you don’t have to implement an email client from the beginning, but you can use an existing one like the default Email app provided from Android, Gmail, Outlook, K-9 Mail etc. For this purpose, we need to write an Activity that launches an email client, using an implicit Intent with the right action and data. In this example, we are going to send an email from our app by using an Intent object that launches existing email clients.

Following section explains different parts of our Intent object required to send an email.

**Intent Object - Action to send Email**

You will use ACTION\_SEND action to launch an email client installed on your Android device. Following is simple syntax to create an intent with ACTION\_SEND action.

Intent emailIntent = new Intent(Intent.ACTION\_SEND);

Intent Object - Data/Type to send Email

To send an email you need to specify mailto: as URI using setData() method and data type will be to text/plain using setType() method as follows −

emailIntent.setData(Uri.parse("mailto:"));

emailIntent.setType("text/plain");

Intent Object - Extra to send Email

Android has built-in support to add TO, SUBJECT, CC, TEXT etc. fields which can be attached to the intent before sending the intent to a target email client.

A CharSequence dialog title to provide to the user when used with a ACTION\_CHOOSER.

Android allows us to integrate google maps in our application. You can show any location on the map , or can show different routes on the map e.t.c. You can also customize the map according to your choices.

**13 Google Map - Layout file**

Now you have to add the map fragment into xml layout file. Its syntax is given below −

<fragment

android:id="@+id/map"

android:name="com.google.android.gms.maps.MapFragment"

android:layout\_width="match\_parent"

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

**Google Map - AndroidManifest file**

The next thing you need to do is to add some permissions along with the Google Map API key in the AndroidManifest.XML file. Its syntax is given below −

<!--Permissions-->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="com.google.android.providers.gsf.permission.

READ\_GSERVICES" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<!--Google MAP API key-->

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0" />

**Customizing Google Map**

You can easily customize google map from its <meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0" />

**Customizing Google Map**

You can easily customize google map from its default view , and change it according to your demand.

**Adding Marker**

You can place a maker with some text over it displaying your location on the map. It can be done by via addMarker() method. Its syntax is given below −

final LatLng TutorialsPoint = new LatLng(21 , 57);

Marker TP = googleMap.addMarker(new MarkerOptions()

.position(TutorialsPoint).title("TutorialsPoint"));

**Changing Map Type**

You can also change the type of the MAP. There are four different types of map and each give a different view of the map. These types are Normal,Hybrid,Satellite and terrain. You can use them as below

googleMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_SATELLITE);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_TERRAIN);

**Enable/Disable zoom**

You can also enable or disable the zoom gestures in the map by calling the setZoomControlsEnabled(boolean) method. Its syntax is given below −

googleMap.getUiSettings().setZoomGesturesEnabled(true);

Apart from these customization, there are other methods available in the GoogleMap class , that helps you more customize the map. They are listed below −

1 addCircle(CircleOptions options)

This method add a circle to the map

2 addPolygon(PolygonOptions options)

This method add a polygon to the map

3 addTileOverlay(TileOverlayOptions options)

This method add tile overlay to the map

4 animateCamera(CameraUpdate update)

This method Moves the map according to the update with an animation

5 clear()

This method removes everything from the map.

6 getMyLocation()

This method returns the currently displayed user location.

7 moveCamera(CameraUpdate update)

This method repositions the camera according to the instructions defined in the update

8 setTrafficEnabled(boolean enabled)

This method Toggles the traffic layer on or off.

9 snapshot(GoogleMap.SnapshotReadyCallback callback)

This method Takes a snapshot of the map

10 stopAnimation()

This method stops the camera animation if there is one in progressdefault view , and change it according to your demand.

**Adding Marker**

You can place a maker with some text over it displaying your location on the map. It can be done by via addMarker() method. Its syntax is given below −

final LatLng TutorialsPoint = new LatLng(21 , 57);

Marker TP = googleMap.addMarker(new MarkerOptions()

position(TutorialsPoint).title("TutorialsPoint"));

**14 What are Android Web Services?**

Android Web Services is a standardised system that helps various applications and systems to communicate with each other. While communicating, they can exchange information and also share some services among themselves. Android web services can run on the internet or private local networks depending on the requirements. Android Web Services are pretty helpful in establishing connections and ensuring security while sharing data in the network.

Below is a ubiquitous example of android web services, which would help you understand the android web services. Suppose your application can seek restaurant data from the server and then display it in your application. The application can then send back the desired list of items to the server, and then the admins can fulfil your order.

So, you can notice that data has to flow from server to application and from application back to the server. In such scenarios, Android Web Services play a significant role in establishing and deciding the protocols for communication

How do Web Servers work?

When we consider web services, there are two essential parts of a web service known as client and server.

* Client: The client is the user or the requesting application that requests data or information from a server.
* Server: Server is like an admin who responds to client’s requests. Servers can handle more than one client and decide whether to fulfil or decline client requests. The server is the place where our web service is hosted globally or locally.

The below figure describes how web servers work and how client and server interact.

android web servers workin

**15 Components of Android Web Services**

Now, let’s see some of the components present in the web server and understand their role.

1. Publisher – Publisher provides web services to clients and is also known as a service provider.

2. Subscriber – The subscriber is the user or the application that requests services from the publisher.

3. Broker – Usually, the Subscriber is unknown about the publisher and needs something to guide the location of the web service. So, the broker is the application that helps the subscriber to identify the web service. The broker gives the subscriber access to UDDI(User descriptive, discovery, and integration).

Now, let’s see the roles and operations each of them carries out.

Publish – Publish means creating the web service and describing its location to the broker for its easy identification by subscribers.

Subscribe – Subscribe means that the subscriber locates the web service with the help of the broker.

Bind – After the subscriber successfully fetches the location, the subscriber binds itself with the web service to exchange information.

**Characteristics of Web Services in Android**

I hope until now you are clear with what web services are and the components involved in them. Now, it’s time for us to look at some of the web services in android.

a. Web Services are XML-Based – Both client and server use XML as their communication language. In other words, the client requests in XML and receives a response, which is XML.I

b. Web services are not tied to one specific operating system or programming language. For example, a Java-based apa. Web Services are XML-Based – Both client and server use XML as their communication language. In other words, the client requests in XML and receives a response, which is XML.

b. Web services are not tied to one specific operating system or programming language. For example, a Java-based application can communicate with a Perl based application.

c. Web Services are available on both the internet or on the local network.

d. Web Services are not tightly coupled. In other words, the client-side web service and the provider side web service are not directly tied.

e. Web Services can be either synchronous or asynchronous. By being synchronous, the clients can directly perform functionalities without establishing a connection. By being asynchronous, the client first needs to establish a connection and then perform the functionalities.

f. Web Services allow you to share multiple files, including documents and complex ones.

**XML Remote Procedure Calls (RPC)**

One of the best ways to communicate documents and information across computers is through remote procedure calls. Let’s look at some XML-RPC details:

1. It does Remote Calls using XML messages.

2. These queries are sent over HTTP POST and are encoded in XML.

3. XML responses are integrated similarly to HTTP responses.

4. XML- RPC is platform-agnostic as well as language-agnostic.

5. It enables several apps to communicate with one another.

6. It was created in accordance with W3C guidelines.

**Types of Web Services in Android**

There are four types of Web Services available in android and are listed below:

1. XML-RPC

XML-RPC, popularly known as Remote Procedure Calls, are used to exchange information among large devices. Every call is encoded using XML, and HTTP is used for its transmission.

2. UDDI

UDDI is an acronym for Universal Descriptive, Discovery, and Integration. It is an XML-based standard that is used to describe, publish, and discover new web services.

3. SOAP

SOAP refers to the Simple Object Access Protocol and is an XML-based web service protocol for exchanging data or documents over HTTP (Hypertext transfer protocol) or SMTP (Simple Message Transfer Protocol). It allows separate processes on different platforms to communicate with one another.

4. REST

REST(REpresentational State Transfer) is an architectural pattern that allows multiple web service-based systems to interact and communicate efficiently. RESTful systems(the system in compliance with REST service) are distinguished by their statelessness and separation of client and server concerns.

**Advantages of Android Web Services**

1. Web services make it possible for various applications to communicate with one another.

2. Reusability is one of the essential benefits of using web services.

3. Web services allow for more efficient communication within and across applications and organisations.

4. They communicate across various apps using a high-quality industry-standard protocol.

5. They employ SOAP over HTTP to enable web services via a low-cost internet connection.

6. Web Services are made available using conventional internet protocols.

7. They enable us to make the functionalities of current programmes available to the public via the internet.

**Limitations of Android Web Services**

Even though web services are pretty beneficial still there are certain demerits of Web Service, which are listed below:

1. They don’t take advantage of new Web advancements.

2. Web services can’t be accessed using a browser.

3. Web services utilise the HTTP protocol, which is unreliable and unsafe.

…videos

**16 Accessing web services**

Welcome to the Java Web Services Tutorial. Here we will learn about web services, useful concepts in web services and then different types of API we have in Java to create web services.

**What is a Web Service**

In simple words, services that can be accessed over network are called web services. So how does it differ from web application, they are also services that are accessed over network. There are few attributes that clarifies this difference.

Web applications are meant for users and to be accessed in browser having human readable format whereas web services are meant for applications to access data in the format of XML, JSON etc.

Web applications always use HTTP/HTTPS protocol whereas traditional web services use SOAP protocol. Recently REST is getting popularity that is an architecture style and almost all times run on HTTP/HTTPS protocol.

Web applications are not meant for reusability whereas this is one of the benefit of web services. A single web service can be used by different kinds of applications.

Web application can access web services to access some data or to perform some tasks, web services can’t access web applications to fetch some data.

Web applications are capable to maintain user session, web services are stateless.

I hope above differences are good enough to clear any confusion with web applications and web services. Both are different concepts and meant for different purpose.

**Types of Web Services**

There are two types of web services.

* SOAP: SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it’s XML based, it’s platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.
* REST: REST is an architectural style for developing web services. It’s getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of Restful web services and they are uniquely identified by their URIs.

**Java Web Services**

Java provides it’s own API to create both SOAP as well as REST web services.

* JAX-WS: JAX-WS stands for Java API for XML Web Services. JAX-WS is XML based Java API to build web services server and client application.
* JAX-RS: Java API for RESTful Web Services (JAX-RS) is the Java API for creating REST web services. JAX-RS uses annotations to simplify the development and deployment of web services.

Both of these APIs are part of standard JDK installation, so we don’t need to add any jars to work with them. Both of these APIs use annotations very heavily.

**Welcome** to the Java Web Services Tutorial. Here we will learn about web services, useful concepts in web services and then different types of API we have in Java to create web services.

**What is a Web Service**

In simple words, services that can be accessed over network are called web services. So how does it differ from web application, they are also services that are accessed over network. There are few attributes that clarifies this difference.

Web applications are meant for users and to be accessed in browser having human readable format whereas web services are meant for applications to access data in the format of XML, JSON etc.

Web applications always use HTTP/HTTPS protocol whereas traditional web services use SOAP protocol. Recently REST is getting popularity that is an architecture style and almost all times run on HTTP/HTTPS protocol.

Web applications are not meant for reusability whereas this is one of the benefit of web services. A single web service can be used by different kinds of applications.

Web application can access web services to access some data or to perform some tasks, web services can’t access web applications to fetch some data.

Web applications are capable to maintain user session, web services are stateless.

I hope above differences are good enough to clear any confusion with web applications and web services. Both are different concepts and meant for different purpose.

**17 Creating services**

A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. A service can essentially take two states −

1 Started

A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

2 Bound

A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

A service has life cycle callback methods that you can implement to monitor changes in the service's state and you can perform work at the appropriate stage. The following diagram on the left shows the life cycle when the service is created with startService() and the diagram on the right shows the life cycle when the service is created with bindService(): (image courtesy : android.com

**18 Threading**

**How Does Threading Work in Android?**

When an application is launched in Android, it creates the primary thread of execution, referred to as the “main” thread. Most thread is liable for dispatching events to the acceptable interface widgets also as communicating with components from the Android UI toolkit. To keep your application responsive, it’s essential to avoid using the most thread to perform any operation which will find yourself keeping it blocked.

Network operations and database calls, also as loading of certain components, are common samples of operations that one should avoid within the main thread. Once they are called within the main thread, they’re called synchronously, which suggests that the UI will remain completely unresponsive until the operation completes. Due to this, tasks requiring calls are usually performed in different threads, which in turn avoids blocking the UI and keeps it responsive while the tasks are being performed. (i.e., they’ve performed asynchronously from the UI).

Android provides some ways of making and managing threads, and lots of third-party libraries exist that make thread management tons more pleasant. However, with numerous approaches at hand, choosing the proper one are often quite confusing. In this article, you’ll study some common scenarios in Android development where threading becomes essential and a few simple solutions which will be applied to those scenarios and more.

**Threading in Android**

In Android, you’ll categorize all threading components into two basic categories:

**Threads that are attached to an activity/fragment:**

These threads are tied to the lifecycle of the activity/fragment and are terminated as soon because the activity/fragment is destroyed.

**Threads that aren’t attached to any activity/fragment:**

These threads can still run beyond the lifetime of the activity/fragment (if any) from which they were spawned.

**O/L Android game developed**

This outlines content is not available next outlines below start

**19 Publishing Android Application**

Android application publishing is a process that makes your Android applications available to users. Infect, publishing is the last phase of the Android application development process.

**Publish**

**Android development life cycle**

Once you developed and fully tested your Android Application, you can start selling or distributing free using Google Play (A famous Android marketplace). You can also release your applications by sending them directly to users or by letting users download them from your own website.

You can check a detailed publishing process at Android official website, but this tutorial will take you through simple steps to launch your application on Google Play. Here is a simplified check list which will help you in launching your Android application −

1. Regression Testing Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2. Application Rating When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3. Targeted Regions Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.
4. Application Size Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.
5. SDK and Screen Compatibility It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.
6. Application Pricing Deciding whether you app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7. Promotional Content It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8. Build and Upload release-ready APK The release-ready APK is what you you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9. Finalize Application Detail Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.