CONSUMING WEB SERVICES USING HTTP

- Creating web service application in android is not a difficult task. We can easily create a restful web service application in android to authenticate or save information into the external database such as oracle, mysql, postgre sql, sql server using other application developed in java, .net, php etc languages. That is what we are going to do.
- Android Restful Web Service
- Before developing web services application, you must have basic knowledge of SOAP and Restful web services.

Resfull Webservices.

RESTful web services are loosely coupled, lightweight **web services** that are particularly well suited for creating APIs for clients spread out across the internet. In the **REST** architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

SOAP Webservices.

- SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.
- SOAP is a recommendation for communication between two applications.
- SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, you will be able to interact with other programming language applications.

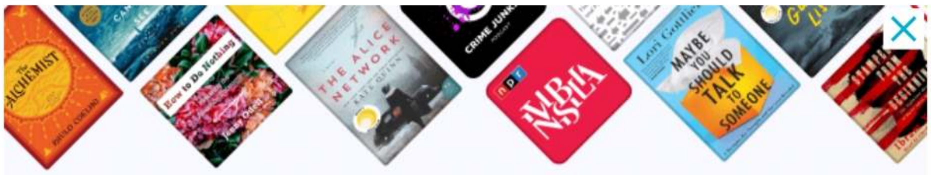
ANDROID RESTFUL WEB SERVICE EXAMPLE

- Using Eclipse, create a new Android project and name it Networking.

Add the following statement in bold to the AndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="net.learn2develop.Networking"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk android:minSdkVersion="14" />
 <uses-permission android:name="android.permission.INTERNET"/>
 <application
 android:icon="@drawable/ic_launcher"
```

```
<activity
 android:label="@string/app_name"
 android:name=".NetworkingActivity">
 <intent-filter >
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 </application>
</manifest>
```



```
package net.learn2develop.Networking;
import android.app.Activity;
import android.os.Bundle;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.util.Log;
public class NetworkingActivity extends Activity {
 /** Called when the activity is first created. */
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 }
}
```

- Define the OpenHttpConnection() method in the NetworkingActivity.java file:

```
public class NetworkingActivity extends Activity {
 private InputStream OpenHttpConnection(String urlString) throws IOException
 {
 InputStream in = null;
 int response = -1;

 URL url = new URL(urlString);
 URLConnection conn = url.openConnection();
```

12:14 PM 75

https://www.scribd.com/presenta

- Define the OpenHttpConnection() method in the NetworkingActivity.java file:

```
public class NetworkingActivity extends Activity {
 private InputStream OpenHttpConnection(String urlString) throws IOException
 {
 InputStream in = null;
 int response = -1;

 URL url = new URL(urlString);
 URLConnection conn = url.openConnection();

 if (!(conn instanceof HttpURLConnection))
 throw new IOException(“Not an HTTP connection”);
 try{
```

Download

```
HttpURLConnection httpConn =
(HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP_OK) {
in = httpConn.getInputStream();
}
}
catch (Exception ex)
{
Log.d("Networking", ex.getLocalizedMessage());
throw new IOException("Error connecting");
}
return in;
}
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
}
```

12:15 PM

HOW IT WORKS

- Because you are using the HTTP protocol to connect to the web, your application needs the INTERNET permission; hence, the fi rst thing you did was add the permission in the AndroidManifest.xml fi le. You then defi ned the OpenHttpConnection() method, which takes a URL string and returns an InputStream object. Using an InputStream object, you can download the data by reading bytes from the stream object. In this method, you made use of the HttpURLConnection object to open an HTTP connection with a remote URL. You set all the various properties of the connection, such as the request method, and so on:

```
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
```

Download

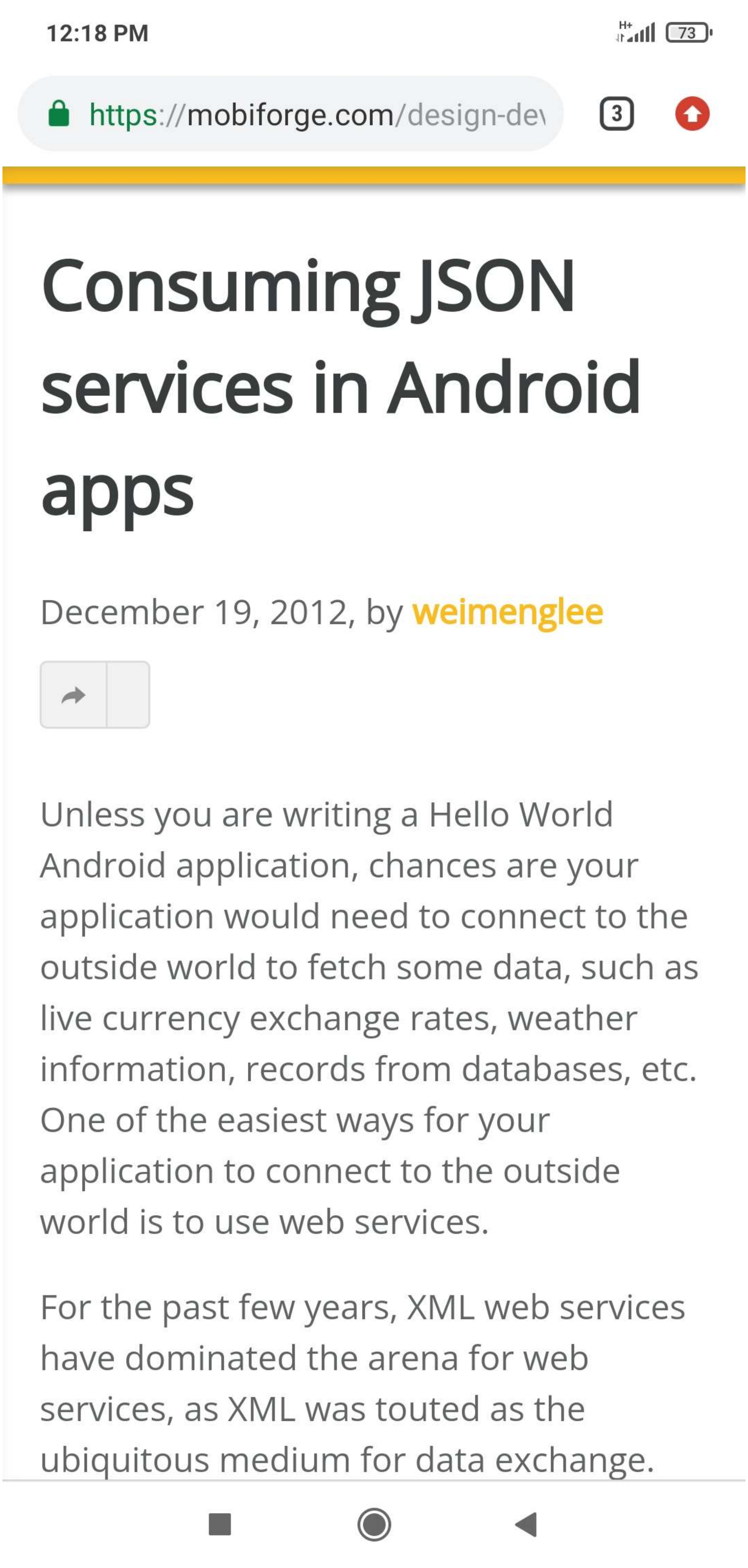
- After trying to establish a connection with the server, the HTTP response code is returned. If the connection is established (via the response code HTTP_OK), then you proceed to get an InputStream object from the connection:

```
httpConn.connect();
 response = httpConn.getResponseCode();
 if (response == HttpURLConnection.HTTP_OK) {
 in = httpConn.getInputStream();
 }
```

ACESSING WEB SERVICES

- SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) for its message format and usually relies on other Application Layer protocols, most notably Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. The following is the structure of the SOAP Envelope:

Example acessing or calling webservice



12:18 PM

https://mobiforge.com/design-de

Consuming JSON services in Android apps

December 19, 2012, by weimenglee

Unless you are writing a Hello World Android application, chances are your application would need to connect to the outside world to fetch some data, such as live currency exchange rates, weather information, records from databases, etc. One of the easiest ways for your application to connect to the outside world is to use web services.

For the past few years, XML web services have dominated the arena for web services, as XML was touted as the ubiquitous medium for data exchange.

world is to use web services.

For the past few years, XML web services have dominated the arena for web services, as XML was touted as the ubiquitous medium for data exchange. However, using XML as the medium for your data payload suffers from the following problems:

1. XML representation is inherently heavy. The use of opening and closing tags add a lot of unnecessary weight to the payload. In the world of mobile applications, shaving a few bytes off the payload will dramatically improve the performance of applications, not to mention the reduction of data transferred over the expensive 3G and LTE wireless networks. This translates into cost savings for both application developers (who need to subscribe to expensive networks for their web servers) and users (who has limited amount of bandwidth to use per subscription).
2. XML representation is difficult to parse. While on the desktop, the DOM (Document Object Model) and SAX (Simple APIs for XML) are the two

web servers) and users (who has limited amount of bandwidth to use per subscription).
2. XML representation is difficult to parse. While on the desktop, the DOM (Document Object Model) and SAX (Simple APIs for XML) are the two commonly used method for parsing XML Documents; on the mobile platform using DOM and SAX are very expensive, both computationally and in terms of memory requirements.

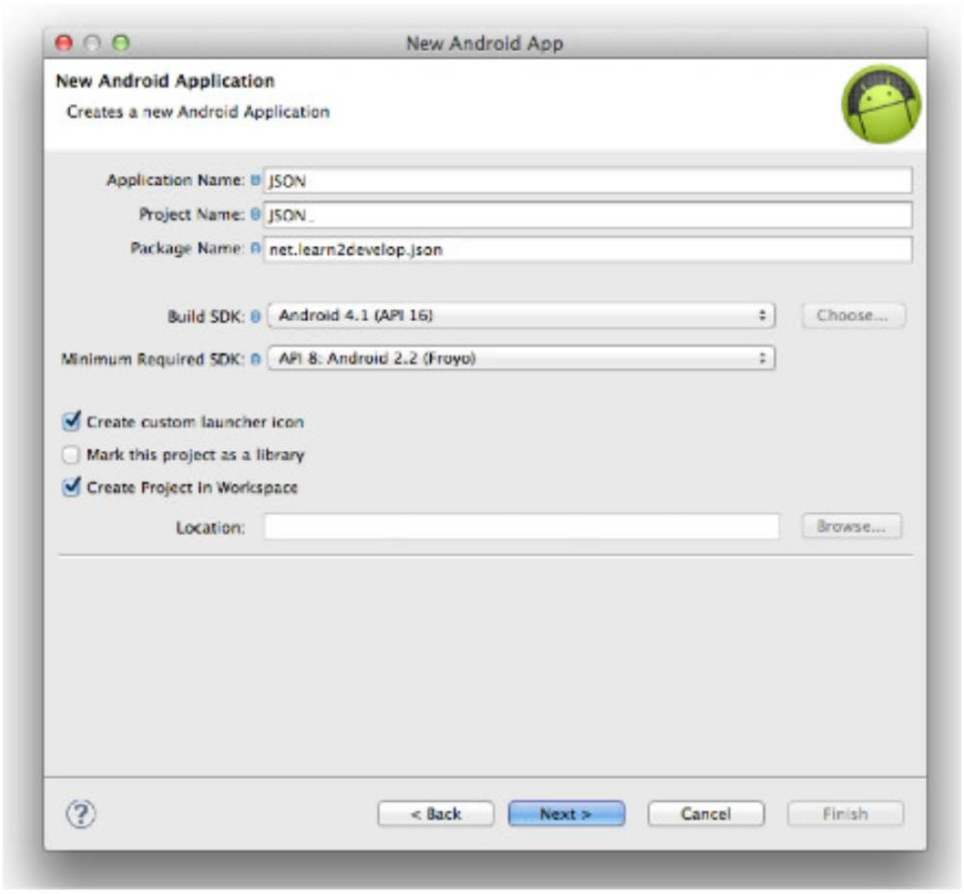
In recent years, another data interchange format has been gaining in popularity – JSON, or JavaScript Object Notation. Like XML, JSON is a text-based open standard for representing data, and it uses characters such as brackets “[{]}”, colon “:” and comma “,”, to represent data. Data are represented using simple key/value pairs, and more complex data are represented as associative arrays.

In this article, I will walk you through on how to consume a JSON service in your Android application.

12:19 PM 73

Creating the Project

For this project, I will be using Eclipse with the Android 4.1 SDK. To start off, create an Android application project and name it as shown in Figure 1.


New Android App

New Android Application

Creates a new Android Application

Application Name: JSON

Project Name: JSON

Package Name: net.learn2develop.json

Build SDK: Android 4.1 (API 16)

Minimum Required SDK: API 8: Android 2.2 (Froyo)

[✓] Create custom launcher icon

[ ] Mark this project as a library

[✓] Create Project in Workspace

Location:

Figure 1: Create Android application project

In the res/layout folder, add in the following statements to the activity_main.xml file:

```
<LinearLayout
```

Figure 1: Create Android application project

In the res/layout folder, add in the following statements to the activity_main.xml file:

```xml
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Latitude" />

    <EditText
        android:id="@+id/txtLat"
        android:layout_width="320dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        android:text="37.77493" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Longitude" />

    <EditText
        android:id="@+id/txtLong"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        android:text="-122.419416"   />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Longitude"  />

    <EditText
        android:id="@+id/txtLong"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        android:text="-122.419416"   />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Weather"
        android:onClick="btnGetWeather"  />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Postal Code"  />

    <EditText
        android:id="@+id/txtPostalCode"
        android:layout_width="320dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="number"
        android:text="89118"  />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Places"
        android:onClick="btnGetPlaces"  />

</LinearLayout>
```

This will create the UI as shown in Figure 2. As you can see, there are actually two parts to the UI:

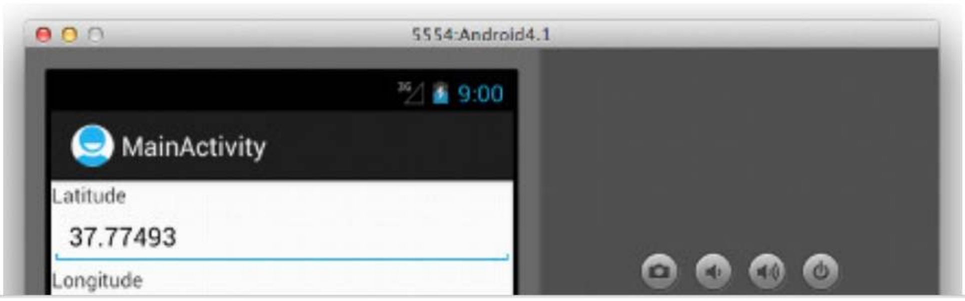
1. The first part allows the user to enter a

```
            android:onClick="btnGetPlaces" />

    </LinearLayout>
```

This will create the UI as shown in Figure 2. As you can see, there are actually two parts to the UI:

1. The first part allows the user to enter a pair of latitude and longitude information. Click on the Get Weather button and you will be able to get information about the weather information for that particular location.
2. The second part allows the user to enter a postal code. Clicking the Get Places button will search for all the towns and cities in the world with this postal code.

In both cases, the data would be retrieved from web services hosted by GeoNames Web Services (http://www.geonames.org/export/web-services.html).



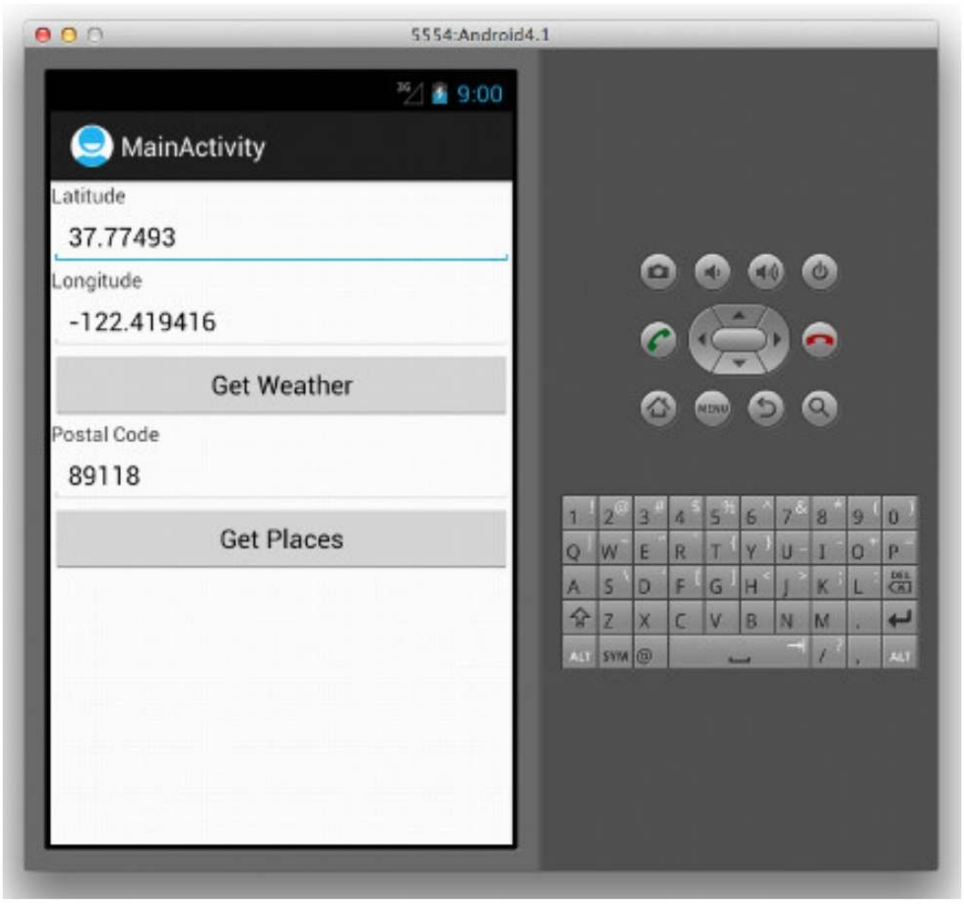


Figure 2: Create the UI

Creating the Helper Method

To connect to a web service, your application needs to first of all connect to the server using HTTP. You need to also determine if you will be using HTTP GET or HTTP POST. Once that is determined, you will fetch the data from the server and get ready for the next step, which is parsing. For this article, the GeoNames Web Services that you will be using uses

Creating the Helper Method

To connect to a web service, your application needs to first of all connect to the server using HTTP. You need to also determine if you will be using HTTP GET or HTTP POST. Once that is determined, you will fetch the data from the server and get ready for the next step, which is parsing. For this article, the GeoNames Web Services that you will be using uses HTTP GET, and hence, you will first of all create the helper method readJSONFeed() in the MainActivity.java file:

```java
package net.learn2develop.json;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

```

in the MainActivity.java file:

```java
package net.learn2develop.json;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {

    public String readJSONFeed(String URL) {
        StringBuilder stringBuilder = new StringBuilder();
        HttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(URL);
        try {
            HttpResponse response = httpClient.execute(httpGet);
            StatusLine statusLine = response.getStatusLine();
            int statusCode = statusLine.getStatusCode();
            if (statusCode == 200) {
                HttpEntity entity = response.getEntity();
                InputStream inputStream = entity.getContent();
                BufferedReader reader = new BufferedReader(
                        new InputStreamReader(inputStream));
                String line;
                while ((line = reader.readLine()) != null) {
```

```
            if (statusCode == 200) {
                HttpEntity entity = response.getEntity();
                InputStream inputStream = entity.getContent();
                BufferedReader reader = new BufferedReader(
                        new InputStreamReader(inputStream));
                String line;
                while ((line = reader.readLine()) != null) {
                    stringBuilder.append(line);
                }
                inputStream.close();
            } else {
                Log.d("JSON", "Failed to download file");
            }
        } catch (Exception e) {
            Log.d("readJSONFeed", e.getLocalizedMessage());
        }
        return stringBuilder.toString();
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

The readJSONFeed() method takes in a string representing the URL of the web service and then connects to the server using HTTP GET. You make use of the HttpClient class to connect to the server, the HttpGet class to specify the URL of the server, and the HttpResponse class to get

```
}
```

The readJSONFeed() method takes in a string representing the URL of the web service and then connects to the server using HTTP GET. You make use of the HttpClient class to connect to the server, the HttpGet class to specify the URL of the server, and the HttpResponse class to get the connection status from the server. Once the connection is established successfully, you all use the BufferedReader and InputStreamReader classes to download the result (which is a JSON string in this example) from the server. The readJSONFeed() method then returns the JSON string.

As you need Internet access for this project to work, remember to add the INTERNET permission in the AndroidManifest.xml file:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.json"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Getting Weather Information

Now that you can connect to the server to download the JSON result, it is now time to connect to the GeoNames Web Services to get weather information. In particular, to get the weather information of a particular location, you will use the following URL (replace the <lat> and <lng> with the actual latitude and longitude):

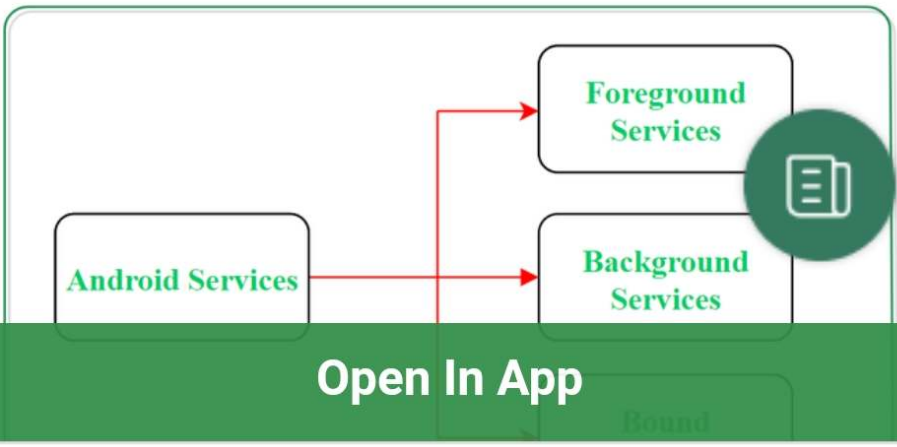
http://ws.geonames.org/findNearByWeatherJSON?lat=<lat&gt&lng=<lng>

A sample result from the server looks like this:

Services in Android with Example

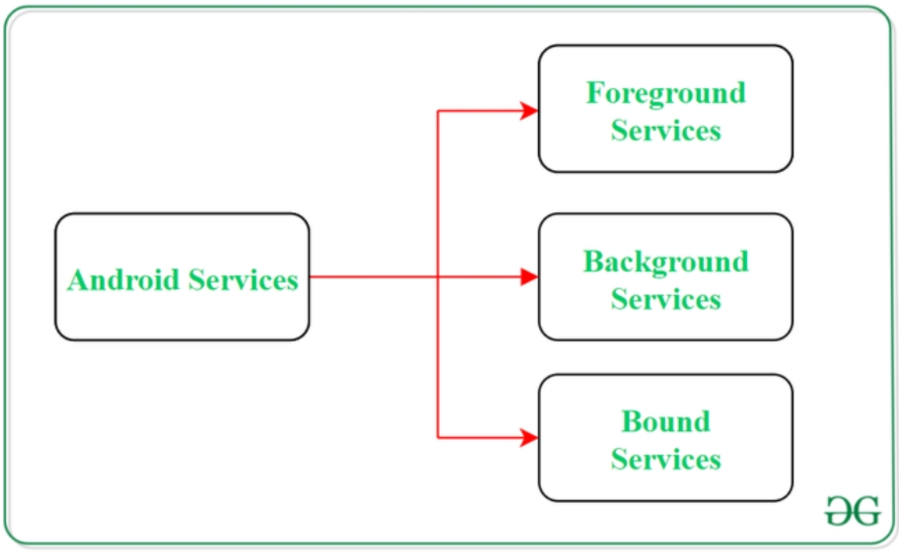
Services in Android are a special component that facilitates an application to run in the background in order to perform long-running operation tasks. The prime aim of a service is to ensure that the application remains active in the background so that the user can operate multiple applications at the same time. A user-interface is not desirable for android services as it is designed to operate long-running processes without any user intervention. A service can run continuously in the background even if the application is closed or the user sw to another application. Further, application components can bind itself to service to carry out inter-process application is closed or the user switches to another application. Further, application components can bind itself to service to carry out inter-process communication(IPC). There is a major difference between android services and threads, one must not be confused between the two. Thread is a feature provided by the Operating system to allow the user to perform operations in the background. While service is an android component that performs a long-running operation about which the user might not be aware of as it does not have UI.

Types of Android Services



android component that performs a long running operation about which the user might not be aware of as it does not have UI.

Types of Android Services



1. Foreground Services:

Services that notify the user about its ongoing operations are termed as Foreground Services. Users can inte with the service by the notifications provided about the ongoing task. Such as in downloading a file, the user can keep

1. Foreground Services:

Services that notify the user about its ongoing operations are termed as Foreground Services. Users can interact with the service by the notifications provided about the ongoing task. Such as in downloading a file, the user can keep track of the progress in downloading and can also pause and resume the process.

2. Background Services:

Background services do not require a user intervention. These services do not notify the user about ongoing background

2. Background Services:

Background services do not require any user intervention. These services do not notify the user about ongoing background tasks and users also cannot access them. The process like schedule syncing of data or storing of data fall under this service.

3. Bound Services:

This type of android service allows the components of the application like activity to bound themselves with it. Bound services perform their task as long as any application component is bound to it. More than one component is allowed to bind themselves with a service at a time. In order to bind an application component with a service **bindService()** method is used.

The Life Cycle of Android Servic

In android, services have 2 possible paths to complete its life cycle namely **Started and Bounded**

The Life Cycle of Android Services

In android, services have 2 possible paths to complete its life cycle namely **Started and Bounded**.

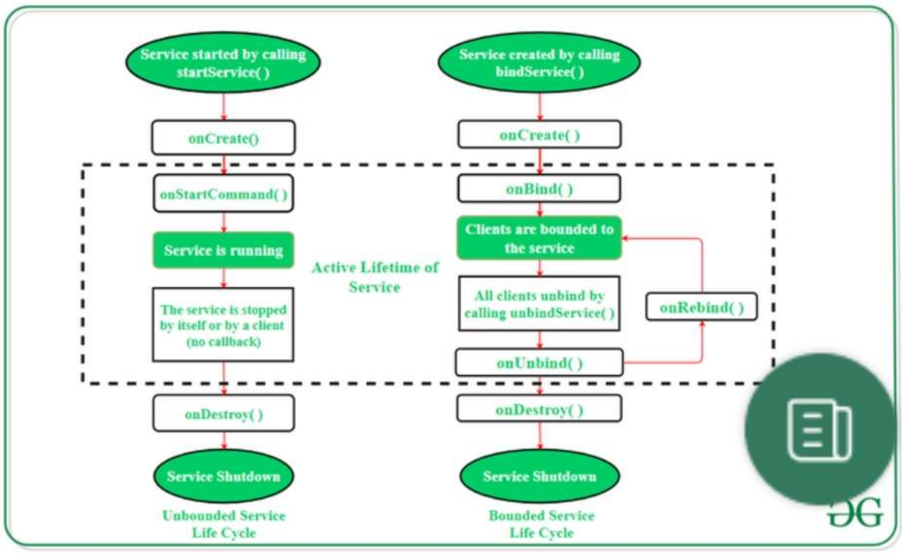
1. Started Service (Unbounded Service):
By following this path, a service will initiate when an application component calls the **startService()** method. Once initiated, the service can run continuously in the background even if the component is destroyed which was responsible for the start of the service. Two option are available to stop the execution of service:

- By calling **stopService()** method,
- The service can stop itself by using **stopSelf()** method.

2. Bounded Service:
It can be treated as a server in a client-server interface. By following this path, android application components can send

2. Bounded Service:

It can be treated as a server in a client-server interface. By following this path, android application components can send requests to the service and can fetch results. A service is termed as bounded when an application component binds itself with a service by calling **bindService()** method. To stop the execution of this service, all the components must unbind themselves from the service by using **unbindService()** method.


Service started by calling startService( )
onCreate()
onStartCommand( )
Service is running
The service is stopped by itself or by a client (no callback)
onDestroy( )
Service Shutdown
Unbounded Service Life Cycle
Active Lifetime of Service
Service created by calling bindService( )
onCreate( )
onBind( )
Clients are bounded to the service
All clients unbind by calling unbindService( )
onRebind( )
onUnbind( )
onDestroy( )
Service Shutdown
Bounded Service Life Cycle

To carry out a downloading task in the background, the ***startService()*** *method will be called. Whereas to get information regarding the download progress and to pause or resume the process while the application is still in the background,* ***the service must be bounded with a component*** *which can perform these tasks.*

Fundamentals of Android Services

A user-defined service can be created through a normal class which is extending the **class Service**. Further, to carry out the operations of service on applications, there are certain callback methods which are needed to be **overridden**. The following are some of the important methods of Android Services:

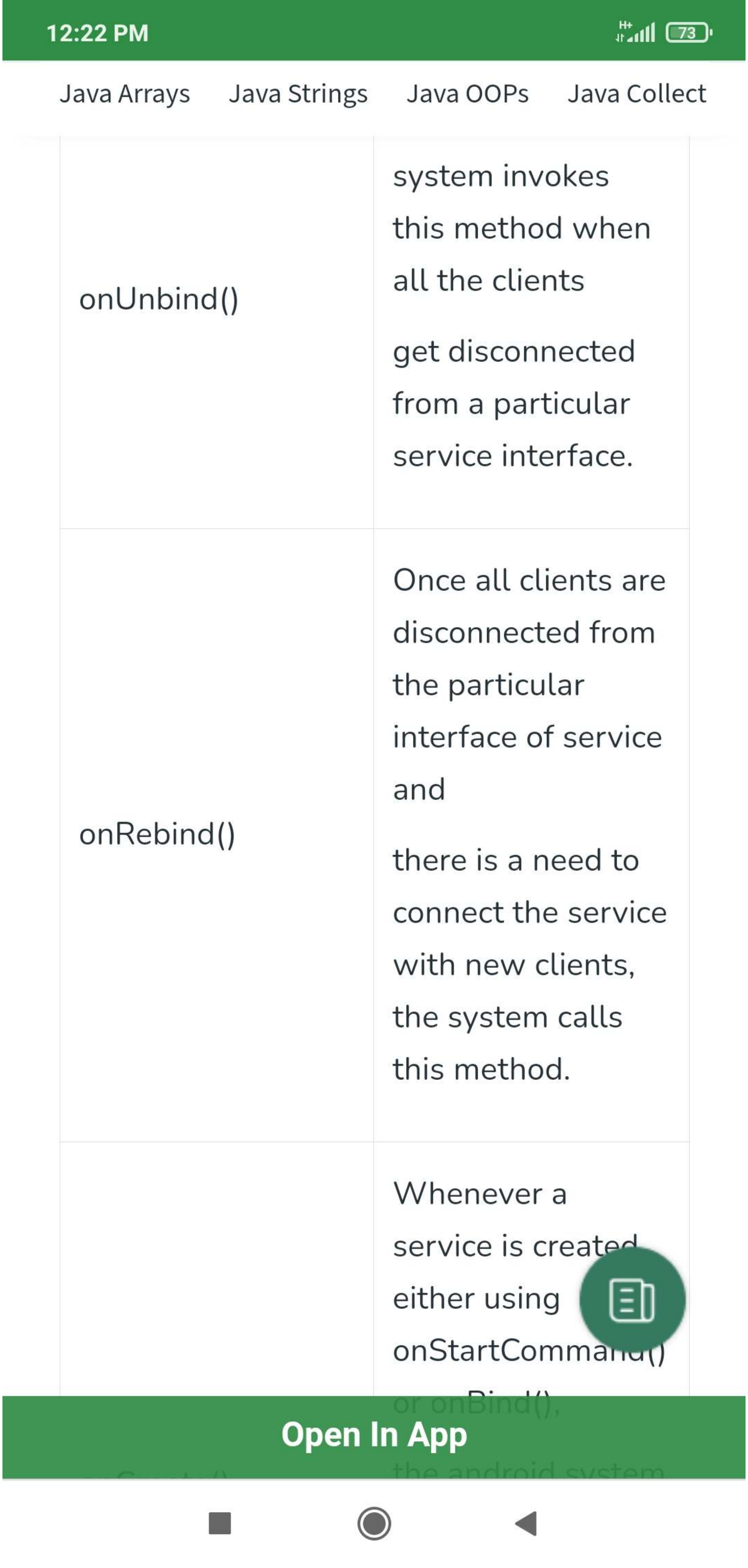
Fundamentals of Android Services

A user-defined service can be created through a normal class which is extending the **class Service**. Further, to carry out the operations of service on applications, there are certain callback methods which are needed to be **overridden**. The following are some of the important methods of Android Services:

Methods	Description
onStartCommand()	The Android service calls this method when a component(eg: activity) requests to s service using startService(). Once the service is

	component(eg: activity)
onStartCommand()	requests to start a service using startService(). Once the service is started, it can be stopped explicitly using stopService() or stopSelf() methods.
	This method is mandatory to implement in android service and is invoked whenever an application component ca the bindServi method in order to itself with a

onBind()	the bindService() method in order to bind itself with a service. User-interface is also provided to communicate with the service effectively by returning an IBinder object. If the binding of service is not required then the method must return null.
onUnbind()	The Android system invokes this method when all the clients get disconnected from a particular service interface.



onUnbind()	system invokes this method when all the clients get disconnected from a particular service interface.
onRebind()	Once all clients are disconnected from the particular interface of service and there is a need to connect the service with new clients, the system calls this method.
onCreate()	Whenever a service is created either using onStartCommand() or onBind(), the android system

onCreate()	Whenever a service is created either using onStartCommand() or onBind(), the android system calls this method. This method is necessary to perform a one-time-set-up.
onDestroy()	When a service is no longer in use, the system invokes this method just before the service destroys as a final clean up call. Services must implement th method in order to clean up resources like registered

	necessary to perform a one-time-set-up.
onDestroy()	When a service is no longer in use, the system invokes this method just before the service destroys as a final clean up call. Services must implement this method in order to clean up resources like registered listeners, threads, receivers, etc.

Binding Activities to Services

When an Activity is bound to a Service, it maintains a reference to the Service instance itself, allowing you to make method calls on the running Service as you would any other instantiated class.

Binding is available for Activities that would benefi t from a more detailed interface with a Service. To support binding for a Service, implement the onBind method as shown in the simple example below:

```
private final IBinder binder = new MyBinder();
@Override
public IBinder onBind(Intent intent) {
return binder;
}
public class MyBinder extends Binder {
MyService getService() {
return MyService.this;
}
}
```

The connection between the Service and Activity is represented as a ServiceConnection. You'll need to implement a new ServiceConnection, overriding the onServiceConnected and onServiceDisconnected methods to get a reference to the Service instance once a connection has been established.

```
// Reference to the service
private MyService serviceBinder;
// Handles the connection between the service and activity
private ServiceConnection mConnection = new ServiceConnection() {
public void onServiceConnected(ComponentName className, IBinder service) {
// Called when the connection is made.
serviceBinder = ((MyService.MyBinder)service).getService();
}
public void onServiceDisconnected(ComponentName className) {
// Received when the service unexpectedly disconnects.
serviceBinder = null;
}
};
```

To perform the binding, call bindService, passing in an Intent (either explicit or implicit) that selects the Service to bind to and an instance of your new ServiceConnection implementation, as shown in this skeleton code:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
// Bind to the service
Intent bindIntent = new Intent(MyActivity.this, MyService.class);
bindService(bindIntent, mConnection, Context.BIND_AUTO_CREATE);
        }
```

Once the Service has been bound, all of its public methods and properties are available through the serviceBinder object obtained from the onServiceConnected handler.

Android applications do not (normally) share memory, but in some cases, your application may want to interact with (and bind to) Services running in different application processes.

You can communicate with a Service running in a different process using broadcast Intents or through the extras Bundle in the Intent used to start the Service. If you need a more tightly coupled connection, you can make a Service available for binding across application boundaries using AIDL. AIDL defi nes the Service's interface in terms of OS level primitives, allowing Android to transmit objects across process boundaries. AIDL defi nitions are covered in Chapter 11.

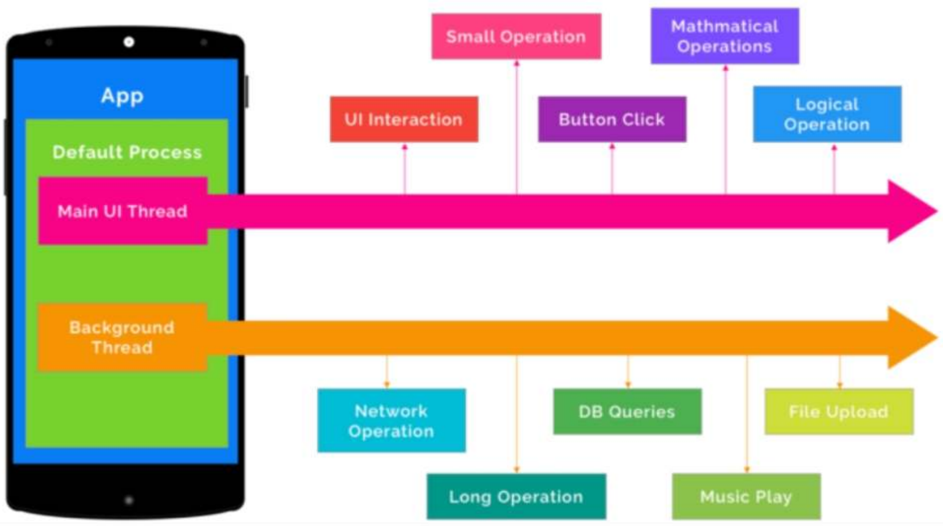
12:29 PM

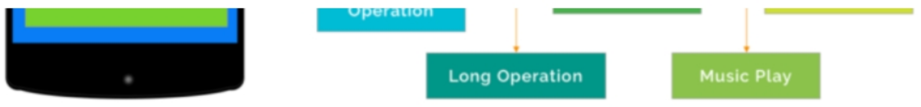
https://www.topcoder.com/thrive/

August 31, 2022

THREAD IN ANDROID

Thread is one of the important concepts in Android. Thread is a lightweight sub-process that provides us a way to do background operations without interrupting the User Interface (UI). When an app is launched, it creates a single thread in which all app components will run by default. The thread which is created by the runtime system is known as the main thread. The main thread's primary role is to handle the UI in terms of event handling and interaction with views in the UI. If there is a task that is time-consuming and that task is run on the main thread, then it will stop other tasks until it gets completed, which in turn may result in displaying a warning "Application is unresponsive" to the user by the operating system. So we need different threads for such tasks and some other tasks.





All threading components belong to one of two basic categories.

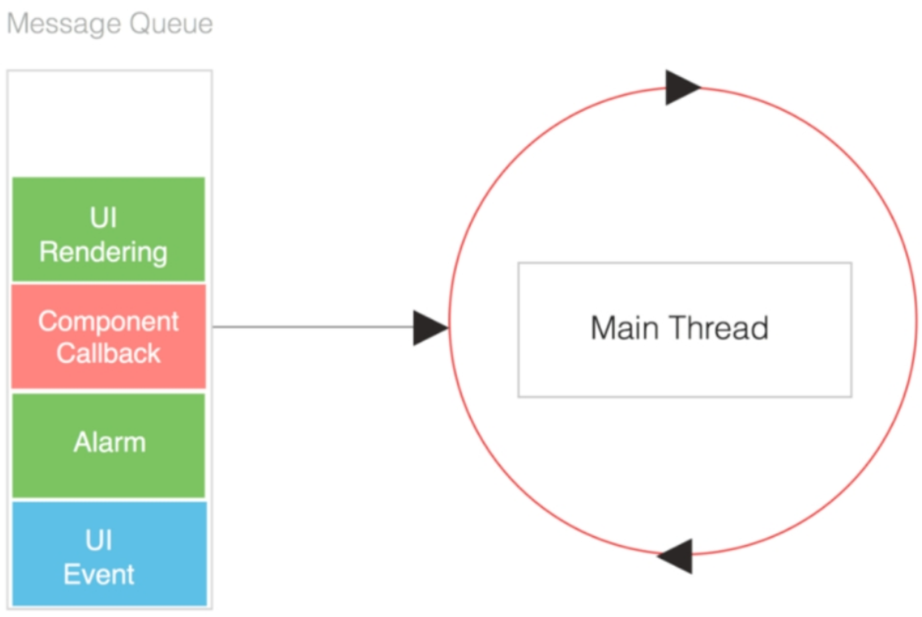
- **The fragment or activity attached threads:** This category of threads are bound to the lifecycle of the activity/fragment and these are terminated as soon as the activity/fragment is destroyed.
  Thread components:
  AsyncTask, Loaders.
- **The fragment or activity not attached threads:** These types of threads can continue to run beyond the lifetime of the activity or fragment from which they were spawned.

Threading Components: Service, Intent Service.

For the two threading components, there are five types of thread used in Android mobile development.

- **Main Thread:** When we launch our app on Android, it creates the first thread of execution called the "Main Thread". The communication between the components from the Android UI toolkit and the dispatching of events to their appropriate UI types of thread used in Android mobile development.

- **Main Thread:** When we launch our app on Android, it creates the first thread of execution called the “Main Thread”. The communication between the components from the Android UI toolkit and the dispatching of events to their appropriate UI widgets is handled by the main thread. We should avoid network operations, database calls, and the loading of certain components in the main thread. Because the main thread is called synchronously when executed, that means the user interface will remain completely unresponsive until the performance completes.



- **UI Thread:** Every app in Android has its own thread which is responsible for running the UI objects, like view objects. Such a thread is known as the UI thread. The UI thread is the main thread of execution for our app as this is where most of the app code is run. The UI thread is where all of our app components (like activities, services, content providers, and broadcast receivers) are created. This thread allows our tasks to perform their background work and then move the results to UI elements such as bitmaps. All objects running on our UI thread will be able to access other objects which are also running on the same UI thread. The tasks that we run on a thread from a thread pool do not run on our UI thread, so they will not have access to UI objects. The data moves from a background thread to the UI thread, using a handler that runs on the UI thread.

- **Worker Thread:** The worker thread is a background thread. The worker threads are created separately, other than threads like the UI thread. As we know from the rules, we cannot block a UI thread so this is where the objects. The data moves from a background thread to the UI thread, using a handler that runs on the UI thread.

- **Worker Thread:** The worker thread is a background thread. The worker threads are created separately, other than threads like the UI thread. As we know from the rules, we cannot block a UI thread so this is where the worker thread comes into play since we can use them to run the child processes and tasks.

- **Any Thread:**

```
@Target([
  AnnotationTarget.FUNCTION, Annot
  AnnotationTarget.PROPERTY_SETTER
  AnnotationTarget.CLASS, Annotati
  AnnotationTarget.VALUE_PARAMETER
])

class AnyThread
```

In Any thread, the annotated method can be called from any thread. If the annotated element is a class, then all methods in the class can be called from Any Thread.

```
})

class AnyThread
```

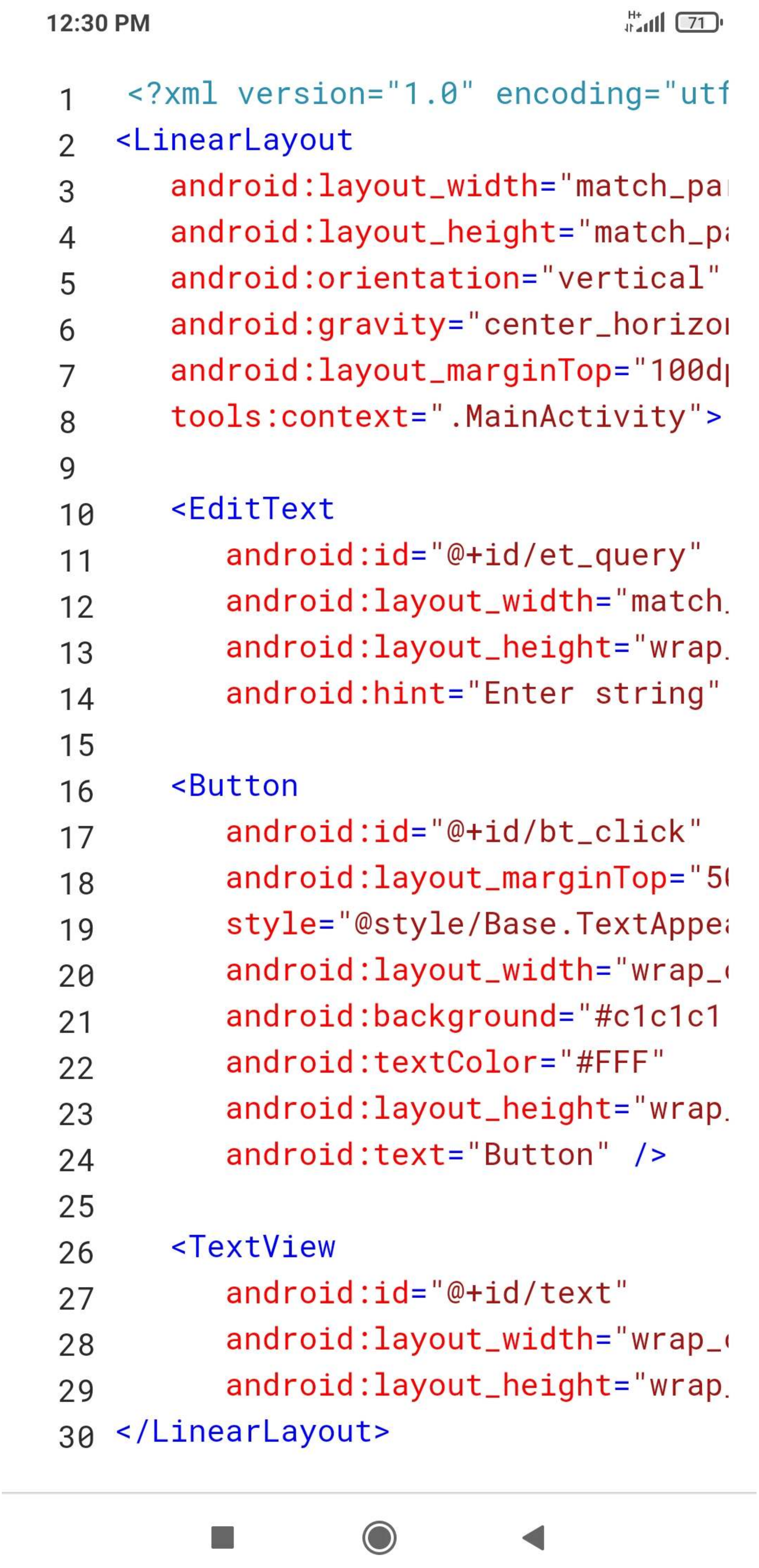
In Any thread, the annotated method can be called from any thread. If the annotated element is a class, then all methods in the class can be called from Any Thread.

- **Binder Thread:** Binder thread represents a separate thread of service. The binder is a mechanism that provides inter-process communication. The binder thread is used in service binding with interprocess communication. This concept is mainly related to service calls with interfaces defined by Android Interface Definition Language (AIDL).

Example

`activity_main.xml`

```
 <?xml version="1.0" encoding="utf
<LinearLayout
    android:layout_width="match_pa
    android:layout_height="match_pa
    android:orientation="vertical"
    android:gravity="center_horizo
    android:layout_marginTop="100d
```



```
<?xml version="1.0" encoding="utf
<LinearLayout
    android:layout_width="match_pa
    android:layout_height="match_p
    android:orientation="vertical"
    android:gravity="center_horizo
    android:layout_marginTop="100d
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/et_query"
        android:layout_width="match
        android:layout_height="wrap
        android:hint="Enter string"

    <Button
        android:id="@+id/bt_click"
        android:layout_marginTop="5
        style="@style/Base.TextAppe
        android:layout_width="wrap_
        android:background="#c1c1c1
        android:textColor="#FFF"
        android:layout_height="wrap
        android:text="Button" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_
        android:layout_height="wrap
</LinearLayout>
```

12:37 PM 70

https://www.upwork.com/resourc

upwork Resource Center

How to Develop an App in 9 Easy Steps (2023 Guide)

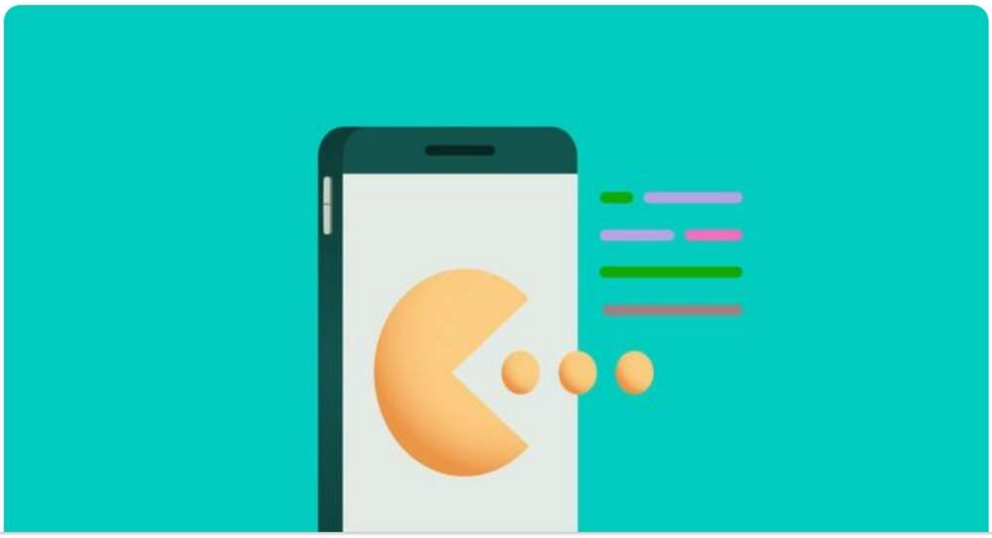
The Upwork Team
Jul 6, 2021 | 13 Min Read

Development & IT Article

Table of Contents

Text Link

Text Link



For a time, businesses generally outsourced mobile app development due to constrained budgets and extended timelines. However, between helpful development platforms and software programs that do much of the groundwork, many small businesses are developing mobile apps in-house.

As a result, more and more companies are embracing the utility of mobile app development for internal purposes or customer use. Although developing an app for the first time can be daunting, it can also be an incredibly rewarding experience. This comprehensive guide will take you through the app development life cycle to help you develop a successful app.

What to identify before developing a mobile app

There are a few considerations to go over before investing time and resources into creating a mobile app. While developing an app is pretty straightforward, planning a strategic digital solution can be complicated.

1. Decide on your operating system

What to identify before developing a mobile app

There are a few considerations to go over before investing time and resources into creating a mobile app. While developing an app is pretty straightforward, planning a strategic digital solution can be complicated.

1. Decide on your operating system

Currently, there are two primary operating systems: iOS and Android. Will your app be a native app, meaning it's developed specifically for a certain OS? Decisions on software compatibility will directly affect app functionality.

A cross-platform framework or an app that works for both iOS and Android will generally be best for future development goals. It's a simple way to set your app up for sustained success, as more users will be able to access it. To help you decide, consider your goals for the app's development.

2. Know your target audience

Make sure you have a good understanding of your target persona. Understand who your target users

2. Know your target audience

Make sure you have a good understanding of your target persona. Understand who your target users are, their goals, their behaviors and preferences, and the platforms and mobile devices they use. This is where your app marketing may come into play. Cater the app to your target audience, to help them enjoy using it and continue to do so.

3. Concept proof your app idea

Before diving into the nitty-gritty and taking those first actionable development steps, consider whether your app is a solution to an existing problem. If it is, think about how it will help. Or, if your app is specific to your organization, consider how the digital solution strategy will help your company's goals.

Pausing to reflect at this stage in the development how the digital solution strategy will help your company's goals.

Pausing to reflect at this stage in the development process can enable you to preview your app's future roadmap. Finalize your highest priorities for the app, and make sure you and your team agree on the top goals. From there, you'll know where to start with a strong footing.

4. Know what's out there

Do your research to make sure your idea isn't already on the market. You don't want to develop your own app only to find that a similar one already exists, rendering yours as less of an original idea than you had hoped.

Make sure your app strategy is unique to you or your company and user-friendly for your target users. For apps that are specific to a business, make sure to check out the competition to grasp an understanding of how your app will stack up against others already on the market.

5. Plan your app design and security

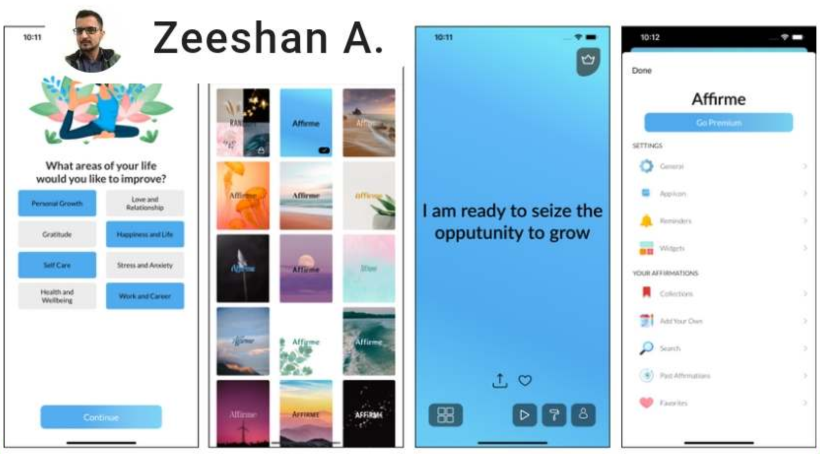
Your app design is key to its success. The program design must have a friendly user interface. The easier it is to use, the more inclined

5. Plan your app design and security

Your app design is key to its success. The program design must have a friendly user interface. The easier it is to use, the more inclined users will be to download the app. Content should also be a top design priority, as should security. People have valuable and potentially sensitive information on their phones. Make sure you consider data protection features and privacy settings. The more advanced your app's security protocols are, the better.

Projects related to this article:

Zeeshan A.



20 day delivery

Browse Project Catalog for more mobile app development services.

How to develop an app in 9 steps

Building an app isn't something that can happen overnight. There are many steps your development team should take to ensure your app launches properly. Keep reading to explore our easy nine-step guide to get you started on developing a new mobile app:

1. Establish a team
2. Conduct competitive research
3. Outline core features
4. Create mockups
5. Plan app security
6. Begin coding
7. Perform multiple tests
8. Gather and implement feedback
9. Launch in the app store

1. Establish a development team

Developing an app has become easier with the help of various software programs and online

8. Gather and implement feedback

9. Launch in the app store

1. Establish a development team

Developing an app has become easier with the help of various software programs and online tools such as app builders. However, any programming language is just that—another language. The app development process is still a real undertaking that is best done with a group of people. A mobile application has a lot of complicated elements that have to come together.

For these reasons, the first step in developing a mobile app is establishing a talented team. To successfully develop an app, you'll greatly benefit from a team of individuals with various backgrounds and expertise.

Building a distributed team can help you develop an app with top independent talent. Instead of being limited to your knowledge or confined by geographic lines, you can pick independent professionals with unique skill sets. To get started, look to Upwork to find the best mobile app developers for your project.

At a minimum, your app development team should include a:

At a minimum, your app development team should include a:

- Product manager: Develops technical spec documents, roadmaps, deadlines, requirements and guides the team.
- UX/UI designer: Designs the graphics, icons and animations. They ensure the app is both engaging and highly intuitive.
- Mobile developer: Codes functionality and integrates APIs, databases and more.
- Quality assurance analyst: Tests the app to make sure it runs smoothly on every device. They're in charge of finding any bugs, UX writing errors and more.
- Digital marketer: Help your app launch successfully by using search engine optimization (SEO) and setting up mobile analytics.

App development teams may even include additional members. For instance, larger teams might have a UX writer, software developer, software engineer, web engineers, and technical writers.

When establishing your team, consider your budget and which areas you need more help to tackle. Hiring freelancers for specific tasks is one

(SEO) and setting up mobile analytics.

App development teams may even include additional members. For instance, larger teams might have a UX writer, software developer, software engineer, web engineers, and technical writers.

When establishing your team, consider your budget and which areas you need more help to tackle. Hiring freelancers for specific tasks is one way to ensure your app is intuitive while sticking to a budget.

2. Conduct competitive research

With more than 2.20 million apps available to Apple users and more than 3.40 million available to Android users, it's important to fully understand what you're up against when it comes to competitor features and customer requirements so you know how to make your product stand out.

Start by researching the market to find apps produced by your competitors. You can outline what your competitors have done right and where they've fallen short. Such research can help you gauge specific insights into what customers like and what you should do differently.

competitor features and customer requirements so you know how to make your product stand out.

Start by researching the market to find apps produced by your competitors. You can outline what your competitors have done right and where they've fallen short. Such research can help you gauge specific insights into what customers like and what you should do differently.

It's also important to understand the market you're joining. You should be able to answer why the market needs your app and what you do differently to solve problems. During this step, you may want to consider talking with potential users. Interviewing customers for user feedback can give you specific insights into their needs. It can also allow your team to develop features that other companies have neglected, giving your app an edge.

3. Outline core features

Next, establish core app features. Now that you understand what's missing from the market and what your target users are searching for, you can develop key features that other development teams have neglected.

Most mobile app developers create a mobile app that's intuitive, easy to navigate, personalizable, teams have neglected.

Most mobile app developers create a mobile app that's intuitive, easy to navigate, personalizable, and simple to use. In addition to this basic foundation, list potential features that can set your app apart from your competitors. Some must-have features include simplicity, speed, and good image resolution. These are all essential features for ensuring a good user experience.

Another feature to consider adding is cross-platform functionality, meaning compatibility with both operating systems. Including a search option is another great utility feature to keep users engaged. While not as useful for game-based applications, providing the ability to search the app is an effective option for e-commerce apps and social media apps.

Finally, consider allowing users to enable push notifications or social media linking. These are other great ways to provide users with relevant and personalized information, keeping them active and engaged over the long haul.

While there are many exciting features to consider as an app maker, don't forget to consider your business's financial capabilities when outlining desired features. For example, while you might want to implement facial recognition for active and engaged over the long haul.

While there are many exciting features to consider as an app maker, don’t forget to consider your business’s financial capabilities when outlining desired features. For example, while you might want to implement facial recognition for login capabilities, this might be expensive to develop. Don’t forget that you can incorporate new features or updates once the app is live.

4. Create mockups

Once requirements have been gathered and key features have been outlined, it’s important to have a user interface (UI) and UX designer develop a mockup, template, and sometimes tutorial of what to expect from the app.

A mockup is a detailed outline of the appearance of the app. Typically, a mockup will follow a cohesive color scheme and typography and includes images, the basic layout and more. When executed correctly, a mockup should give the development team a glimpse into how the app should look and operate.

The advantages of a mockup include:

- Allows the development team to revise the app’s appearance.

executed correctly, a mockup should give the development team a glimpse into how the app should look and operate.

The advantages of a mockup include:

- Allows the development team to revise the app's appearance.
- If you're seeking potential investors, it shows them before the development team begins coding.
- Explains the expectations for the development team.

5. Plan great app security

This next step in building apps is among the most important features your app can have: security. Preventing cybercriminals from stealing user data is paramount. A single breach of the app could cost your company a loss of users and potentially millions of dollars.

Ensure the mobile app and mobile platform are both secure through:

- **Encrypted data:** It's a good idea to use proper encryption of sensitive personal data scattered throughout your app's software. Proper security steps include encryption of the local database, cache, or API communication.

millions of dollars.

Ensure the mobile app and mobile platform are both secure through:

- **Encrypted data:** It's a good idea to use proper encryption of sensitive personal data scattered throughout your app's software. Proper security steps include encryption of the local database, cache, or API communication.
- **Authorized APIs:** Application programming interfaces (APIs) are an essential part of the backend of programming development. Make sure the APIs you use for your application meet the verification standards for the platform that your app is on.
- **Strong authentication:** Ensure the app employs the correct cryptographic key management and appropriate user session authorization—or tokens. Tokens are often assigned to each device and contain different expiration times for sessions.
- **Tamper-detection software:** To stop hackers in their tracks, consider including mobile-specific security features like tamper-detection software and other third-party software. For example, interprocess communication (IPC) is a safety measure that enables communication among other apps and systems. There are many other Apple- and Android-specific software and other UI security features that can help with anti-tampering

security features like tamper-detection software and other third-party software. For example, interprocess communication (IPC) is a safety measure that enables communication among other apps and systems. There are many other Apple- and Android-specific software and other UI security features that can help with anti-tampering tactics.

- **Constant testing for potential breaches:** Most importantly, make sure to test for breaches constantly. Throughout the entire development process, consistently review your code. Identify potential security flaws before a hacker does when the app goes live.

6. Begin coding

There are a few components to consider as you start coding. First, there is both the front end and back end to code. Front-end development refers to the “face” of the app—what the end-user will see. Back-end development is about the “behind the scenes” code, which dictates how the app functions.

Suppose you’re coordinating a team of multiple developers (e.g., some working on the front end and others on the back end). You’ll want to coordinate work processes to ensure a cohesive end product. Using an Agile methodology of to the “face” of the app—what the end-user will see. Back-end development is about the “behind the scenes” code, which dictates how the app functions.

Suppose you’re coordinating a team of multiple developers (e.g., some working on the front end and others on the back end). You’ll want to coordinate work processes to ensure a cohesive end product. Using an Agile methodology of project management can be useful, allowing for efficient, adaptable and flexible coding.

Further, have your team code in a test environment. Setting up an appropriate test environment to check the software’s execution is critical to ensuring a successful final app. Test environment considerations include the database server, front-end environment, operating system and network. You can also designate a bug reporting tool to ensure accurate and granular test data.

7. Perform multiple tests

Although it’s tempting to skip rigorous testing when the project is on a tight budget, quality assurance (QA) is one of the key pieces in developing a successful app. Since app development is so competitive, it’s important to perform QA throughout the entire development perform QA throughout the entire development process. This way, your team can identify any bugs and quickly make improvements to the app before going live to customers.

Some things the QA team should think about when testing include:

- **Front-end vs. back-end functionality:** For the front end, does the mobile app look like it's supposed to from the user side of things? For the back end, does the app function as it's supposed to? For example, if a pop-up message is supposed to disappear when the user clicks on it, does it? Is the little "X" to click it away properly displayed?
- **Device compatibility:** The app needs testing on whatever operating system it's meant for (iOS versus Android, or both). Further, the QA team should confirm compatibility with different versions of operating systems (e.g., Android 7.0 versus Android 10.0). There are also device-specific considerations, like if the app display fits the screen size.
- **App integration:** If the app's core function has interaction with other features, such as the phone's camera or another app like Google Maps, is this integration functional?
- **Application type:** If the app's purpose is to work as both a mobile and web app (making it a "hybrid" app), it needs to be tested for the full range of

interaction with other features, such as the phone's camera or another app like Google Maps, is this integration functional?

- Application type: If the app's purpose is to work as both a mobile and web app (making it a "hybrid" app), it needs to be tested for the full range of functionality across both platforms.
- Installation and storage: Does the app download correctly to the intended device and operating system? Also, keep an eye on app size. An overly large app will take up a lot of room on the end user's phone and may deter them from downloading the app at all.
- Security optimization: Mobile app security is a hot topic. Check safety by ensuring secure source code, performing penetration testing and conducting input validation. Additional steps like confirming the implementation of HTTPS and SSL/TLS security layers are also advisable.

All in all, the QA step helps teams ensure the app is ready to enter the market.

8. Gather and implement user feedback

After rigorous testing, the app should pass

8. Gather and implement user feedback

After rigorous testing, the app should pass inspection by a test group user before launching the app. Although the development team should have based features on customer needs, having someone who doesn't know the app can help provide invaluable user feedback to ensure it is ready for many users with different experience levels.

Testing the app with various users should give your development team a better understanding of what they want and if the app is matching expectations. It should help the team adjust key features to fit a diverse group of users better before the app goes live.

Once the app is live, it's a good idea to continue receiving feedback so you can make necessary updates and adjustments as needed. One way you can gain feedback after the launch of the app is by looking at analytics. It can help your development team understand customer behavior and identify any confusing areas for modification. This kind of insight can also help the team make adjustments to the application before the market launch.

9. Launch in the app store

looking at analytics. It can help your development team understand customer behavior and identify any confusing areas for modification. This kind of insight can also help the team make adjustments to the application before the market launch.

9. Launch in the app store

Lastly comes publication to the app store. The regulations that your app needs to follow depend greatly on the app store where you're applying. Different app stores have different requirements for apps submitted to them.

Developers may only focus on releasing their app to the Google Play Store or Apple App Store. Focusing on a single platform can simplify the app development process because your team only needs to develop one app.

On the other hand, developing an app just for Google Play or Apple can limit your reach. Both app stores have drawbacks; however, developing a cross-platform app ensures good visibility and the potential for more users.

FAQs for mobile app development

How much does it cost to create a mobile app?

receiving feedback so you can make necessary updates and adjustments as needed. One way you can gain feedback after the launch of the app is by looking at analytics. It can help your development team understand customer behavior and identify any confusing areas for modification. This kind of insight can also help the team make adjustments to the application before the market launch.

9. Launch in the app store

Lastly comes publication to the app store. The regulations that your app needs to follow depend greatly on the app store where you're applying. Different app stores have different requirements for apps submitted to them.

Developers may only focus on releasing their app to the Google Play Store or Apple App Store. Focusing on a single platform can simplify the app development process because your team only needs to develop one app.

On the other hand, developing an app just for Google Play or Apple can limit your reach. Both app stores have drawbacks; however, developing a cross-platform app ensures good visibility and the potential for more users.

Android | How to add Radio Buttons in an Android Application?

Android **radio button** is a widget that can have more than one option to choose from. The user can choose only one option at a time. Each option here refers to a radio button and all the options for the topic are together referred to as Radio Group. Hence, Radio Buttons are used inside a RadioGroup.

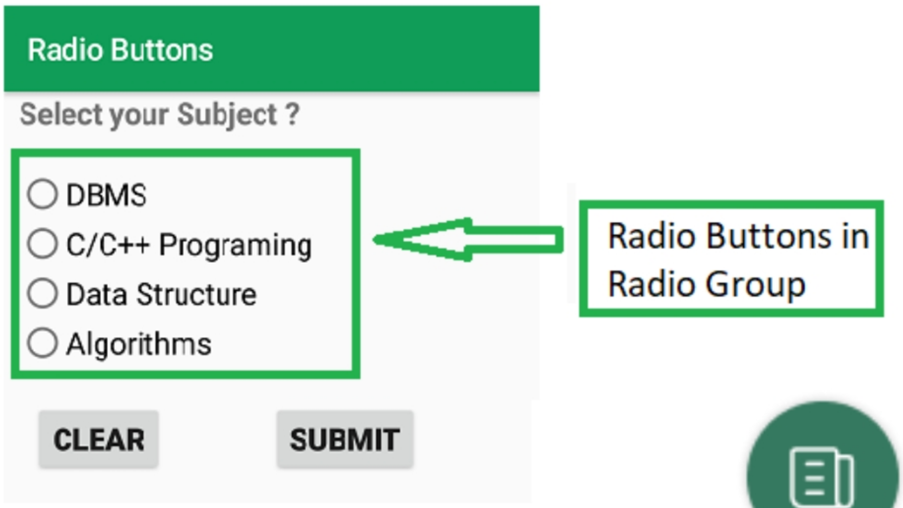
Pre-requisites:

- Android App Development Fundamentals for Beginners
- Guide to Install and Set up Android Studio
- Android | Starting with the first app/android project
- Android | Running your first Android

Pre-requisites:

- Android App Development Fundamentals for Beginners
- Guide to Install and Set up Android Studio
- Android | Starting with the first app/android project
- Android | Running your first Android app

For Example:

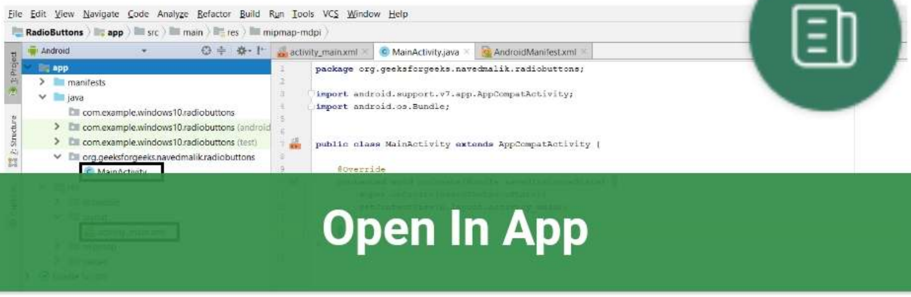


This image shows 4 options of the subjects for a question. In this, each mentioned subject is a Radio Button and all the 4 subjects together are enclosed in a Radio Group.

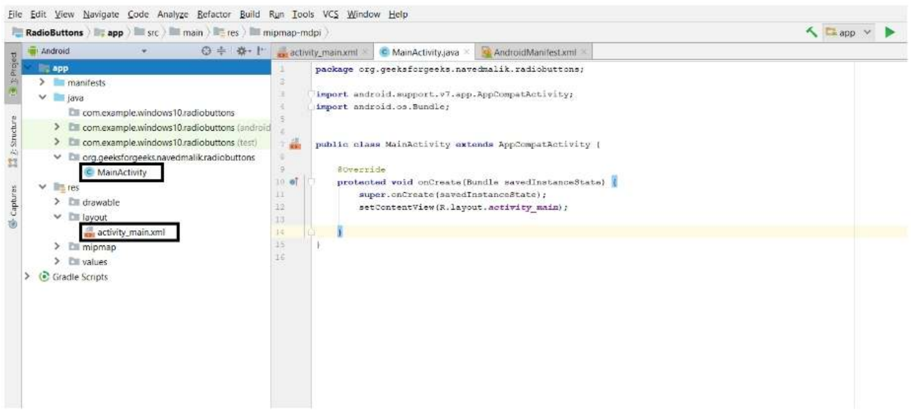
How to create an Android App to use Radio Buttons

This example will help in developing an Android App that creates Radio Buttons according to the above example:

Step 1: First create a new Android Application. This will create an XML file "activity_main.xml" and a Java File "MainActivity.Java". Please refer the pre-requisites to learn more about this step.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
RadioButtons app src main res mipmap-mdpi
Android
app
manifests
java
com.example.windows10.radiobuttons
com.example.windows10.radiobuttons (android
com.example.windows10.radiobuttons (test)
org.geeksforgeeks.navedmalik.radiobuttons
MainActivity
activity_main.xml
MainActivity.java
AndroidManifest.xml

```
package org.geeksforgeeks.navedmalik.radiobuttons;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;


public class MainActivity extends AppCompatActivity {

    @Override
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
RadioButtons app src main res mipmap-mdpi
Android
app
manifests
java
com.example.windows10.radiobuttons
com.example.windows10.radiobuttons (android
com.example.windows10.radiobuttons (test)
org.geeksforgeeks.navedmalik.radiobuttons
MainActivity
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
activity_main.xml
MainActivity.java
AndroidManifest.xml

```
package org.geeksforgeeks.navedmalik.radiobuttons;

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);

    }
}
```

Step 2: Open the "activity_main.xml" file and add the following widgets in a Relative Layout:

- A **TextView** to display the question message
- A **RadioGroup** to hold the option Radio Buttons which are the possible answers
- **4 RadioButtons** to hold an answer each.
- A Submit and a Clear button to s the response.

Open In App

Also, Assign the ID to each of the

- A **TextView** to display the question message
- A **RadioGroup** to hold the option Radio Buttons which are the possible answers
- **4 RadioButtons** to hold an answer each.
- A Submit and a Clear button to store the response.

Also, Assign the **ID** to each of the components along with other attributes as shown in the given image and the code below. The assigned ID on a component helps that component to be easily found and used in the Java files.

Syntax:

```
android:id="@+id/id_name"
```

Here the given IDs are as follows:

- RadioGroup: groupradio
- RadioButton1: radia_id1
- RadioButton2: radia_id2

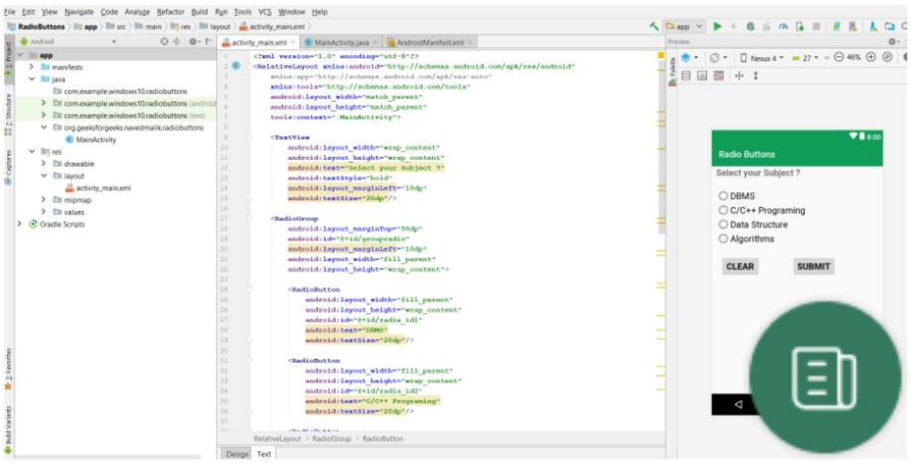
Syntax:

```
android:id="@+id/id_name"
```

Here the given IDs are as follows:

- RadioGroup: groupradio
- RadioButton1: radia_id1
- RadioButton2: radia_id2
- RadioButton3: radia_id3
- RadioButton4: radia_id4
- Submit Button: submit
- Clear Button: clear

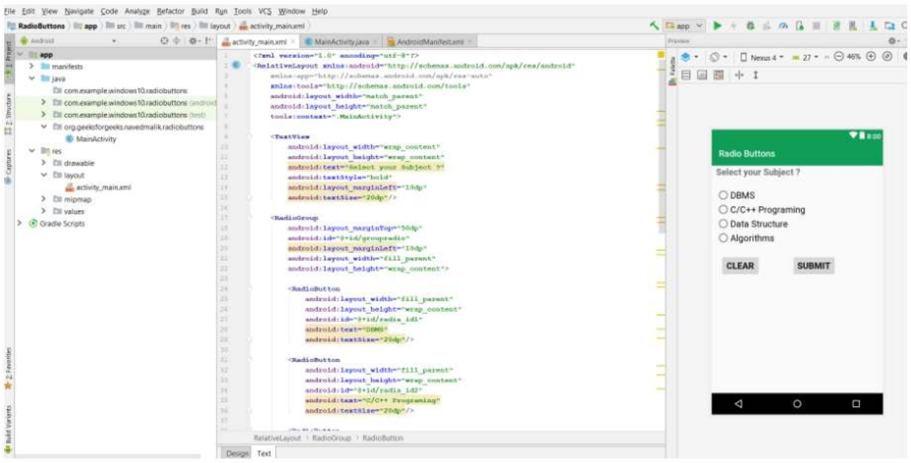
This will make the UI of the Application.



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select your Subject ?"
        android:textStyle="bold"
        android:layout_marginLeft="10dp"
        android:textSize="20dp"/>

    <RadioGroup
        android:layout_marginTop="50dp"
        android:id="@+id/groupradio"
        android:layout_marginLeft="10dp"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <RadioButton
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/radia_id1"
            android:text="DBMS"
            android:textSize="20dp"/>

        <RadioButton
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/radia_id2"
            android:text="C/C++ Programing"
            android:textSize="20dp"/>
```

Open In App

This will make the UI of the Application.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
RadioButtons app src main res layout activity_main.xml
activity_main.xml
MainActivity.java
AndroidManifest.xml
app
manifests
java
com.example.windows10.radiobuttons
org.geeksforgeeks.navedmalik.radiobuttons
MainActivity
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
Radio Buttons
Select your Subject ?
DBMS
C/C++ Programing
Data Structure
Algorithms
CLEAR
SUBMIT
RelativeLayout › RadioGroup › RadioButton
Design
Text

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select your Subject ?"
        android:textStyle="bold"
        android:layout_marginLeft="10dp"
        android:textSize="20dp"/>

    <RadioGroup
        android:layout_marginTop="50dp"
        android:id="@+id/groupradio"
        android:layout_marginLeft="10dp"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <RadioButton
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/radia_id1"
            android:text="DBMS"
            android:textSize="20dp"/>

        <RadioButton
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/radia_id2"
            android:text="C/C++ Programing"
            android:textSize="20dp"/>
```

Step 3: Now, after the UI, this step will create the Backend of Application. For this, open the "MainActivity.java" file and instantiate the components made in the XML file (RadioGroup, TextView, Clear, and Submit Button) using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax: General

and Submit Button) using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax: General

```
ComponentType object =
(ComponentType)findViewById(R.i
d.IdOfTheComponent);
```

Syntax: Components used

```
Button submit =
(Button)findViewById(R.id.submi
t);
Button clear =
(Button)findViewById(R.id.clear
);
RadioGroup radioGroup =
(RadioGroup)findViewById(R.id.g
roupradio);
```

Step 4: This step involves setting u operations on the RadioGroup, RadioButtons, and the Submit and Clear Buttons.

```
(RadioGroup)findViewById(R.id.g
roupradio);
```

Step 4: This step involves setting up the operations on the RadioGroup, RadioButtons, and the Submit and Clear Buttons.

These operations are as follows:

- Unset all the Radio Buttons initially as the default value. This is done by the following command:

```
radioGroup.clearCheck();
```

- Add the Listener on the RadioGroup. This will help to know whenever the user clicks on any Radio Button, and the further operation will be performed. The listener can be added as follows:

```
radioGroup.setOnCheckedChang
Listener(new
RadioGroup.OnCheckedChangeLis
```

```
radioGroup.setOnCheckedChange
Listener(new
RadioGroup.OnCheckedChangeLis
tener(){}
```

- Define the operations to be done when a radio button is clicked. This involves getting the specific radio button that has been clicked, using its id. Then this radio button gets set and the rest of the radio button is reset.
- Add the listener on Submit button and clear button. This will be used to check when the user clicks on the button. This is done as follows:

```
submit.setOnClickListener(new
View.OnClickListener() {}
clear.setOnClickListener(new
View.OnClickListener() {}
```

- In the Submit Button Listener, set the operations to be performed. This

```
submit.setOnClickListener(new
View.OnClickListener() {}
clear.setOnClickListener(new
View.OnClickListener() {}
```

- In the Submit Button Listener, set the operations to be performed. This involves displaying the marked answer in the form of **Toast**.
- In the Clear Button Listener, set the operations to be performed. This involves resetting all the radio buttons.

Step5: Now run the app and operate as follows:

- When the app is opened, it displays a question with 4 answers and a clear and submit button.
- When any answer is clicked, that radio button gets set.
- Clicking on any other radio butto that one and resets the others.
- Clicking on Submit button displays the currently marked answer as a Toast.

operations to be performed. This involves displaying the marked answer in the form of **Toast**.
- In the Clear Button Listener, set the operations to be performed. This involves resetting all the radio buttons.

Step5: Now run the app and operate as follows:

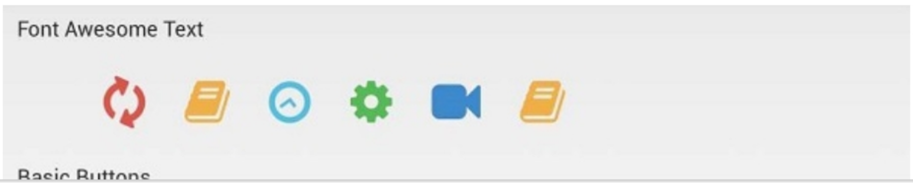
- When the app is opened, it displays a question with 4 answers and a clear and submit button.
- When any answer is clicked, that radio button gets set.
- Clicking on any other radio button sets that one and resets the others.
- Clicking on Submit button displays the currently marked answer as a Toast.
- Clicking on Clear button resets all the radio buttons to their default state.

The complete code of MainActivity.java and activity_main.xml of RadioButt below as follows:

Filename: activity_main.xml

Android - ImageButton Control

An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.



Android button style set

ImageButton Attributes

Following are the important attributes related to ImageButton control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.ImageView** Class –

Sr.No	Attribute & Description
1	**android:adjustViewBounds** Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
2	**android:baseline** This is the offset of the baseline within this view.
3	**android:baselineAlignBottom** If true, the image view will be baseline aligned with based on its bottom edge.

Sr.No	Attribute & Description
1	**android:adjustViewBounds** Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
2	**android:baseline** This is the offset of the baseline within this view.
3	**android:baselineAlignBottom** If true, the image view will be baseline aligned with based on its bottom edge.
4	**android:cropToPadding** If true, the image will be cropped to fit within its padding.
5	**android:src** This sets a drawable as the content of this ImageView.

Inherited from **android.view.View** Class –

Sr.No	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
	android:id

5	This sets a drawable as the content of this ImageView.

Inherited from **android.view.View** Class –

Sr.No	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility** This controls the initial visibility of the view.

Example

This example will take you through simple steps to show how to create your own Android application using Linear Layout and ImageButton.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as myapplication under a package com.example.myapplication as explained in the Hello World Example chapter.
2	Modify src/MainActivity.java file to add a click event.
3	Modify the default content of res/layout/activity_main.xml file to include Android UI control.
4	No need to define default constants in android, Android studio takes care of default constants.
5	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.myapplication/MainActivity.java**. This file can include each of the fundamental

12:44 PM

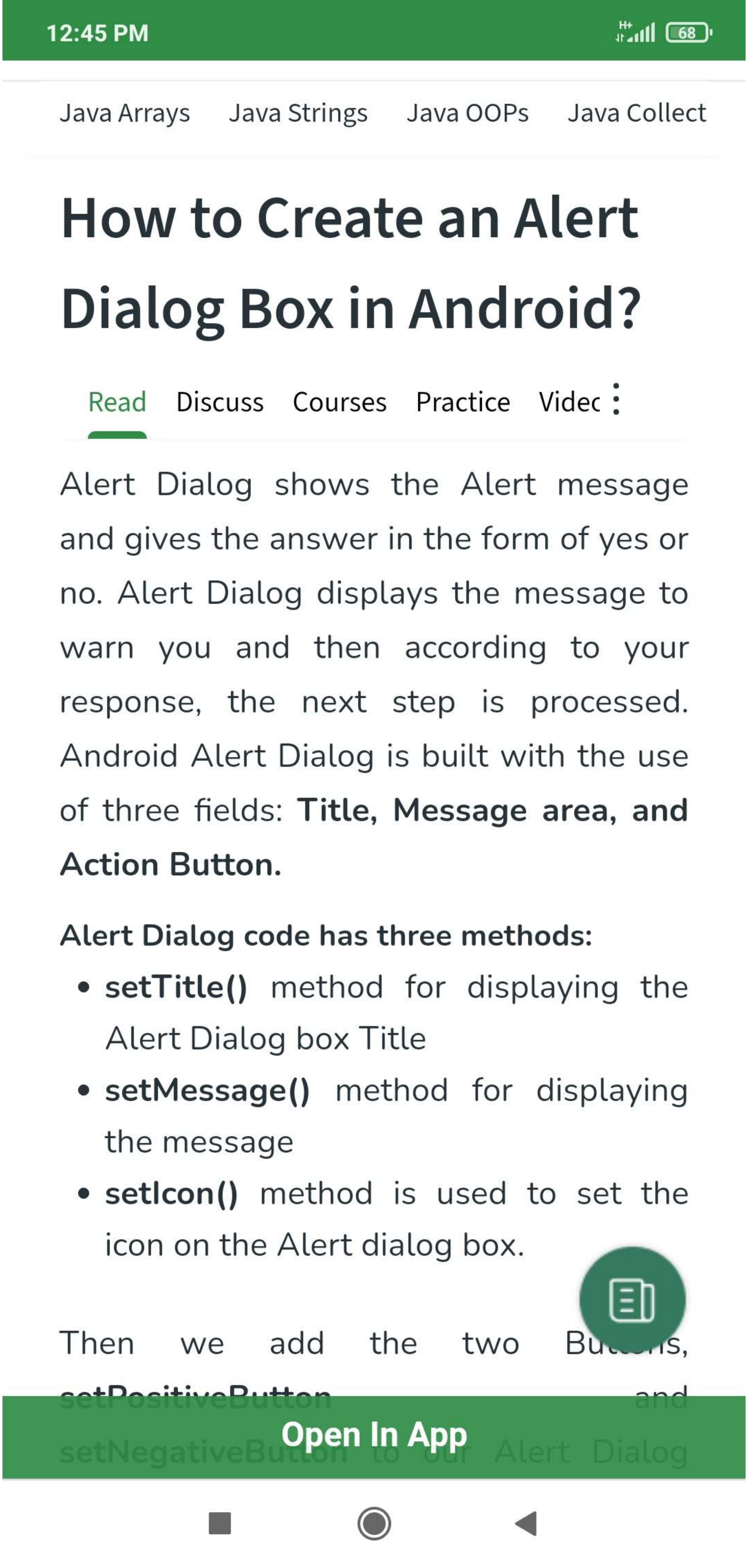
Search tutorials, courses ar

```
package com.example.myapplication

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickL
import android.widget.ImageButton
import android.widget.Toast;

public class MainActivity extends
    ImageButton imgButton;

    @Override
    protected void onCreate(Bundle
        super.onCreate(savedInstanc
        setContentView(R.layout.act

        imgButton =(ImageButton)fin
        imgButton.setOnClickListene
            @Override
            public void onClick(View
                Toast.makeText(getApp
                    resumed",Toast.LEN
            }
        });
    }
}
```

12:44 PM 68

Search tutorials, courses ar

```
tener;



ctivity {



avedInstanceState) {
tate);
ity_main);

iewById(R.id.imageButton);
new View.OnClickListener() {

) {
cationContext(),"You download is
H_LONG).show();
```



How to Create an Alert Dialog Box in Android?

Alert Dialog shows the Alert message and gives the answer in the form of yes or no. Alert Dialog displays the message to warn you and then according to your response, the next step is processed. Android Alert Dialog is built with the use of three fields: **Title, Message area, and Action Button.**

Alert Dialog code has three methods:

- **setTitle()** method for displaying the Alert Dialog box Title
- **setMessage()** method for displaying the message
- **setIcon()** method is used to set the icon on the Alert dialog box.

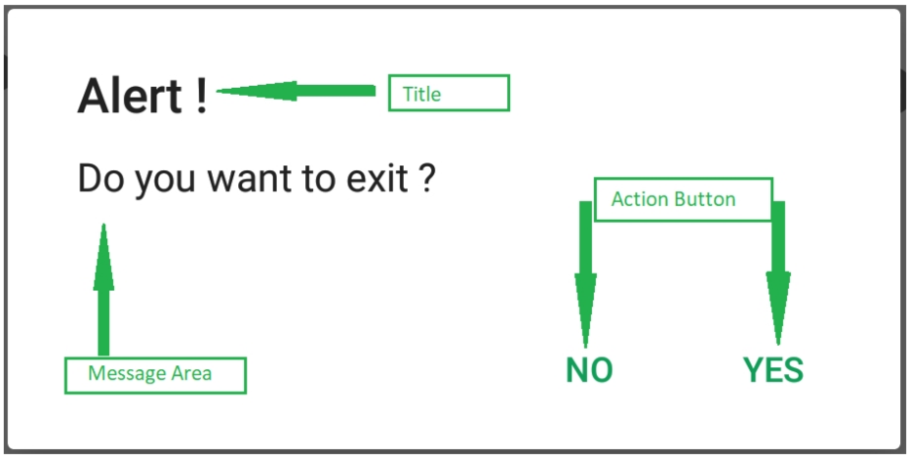
Then we add the two Bu ns, setPositiveButton and setNegativeButton to our Alert Dialog

Open In App

- **setMessage()** method for displaying the message
- **setIcon()** method is used to set the icon on the Alert dialog box.

Then we add the two Buttons, **setPositiveButton** and **setNegativeButton** to our Alert Dialog Box as shown below.

Example:



Step By Step Implementation

Step By Step Implementation

Step 1: Create a New Project in Android Studio

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. The code for that has been given in both **Java and Kotlin Programming Language for Android.**

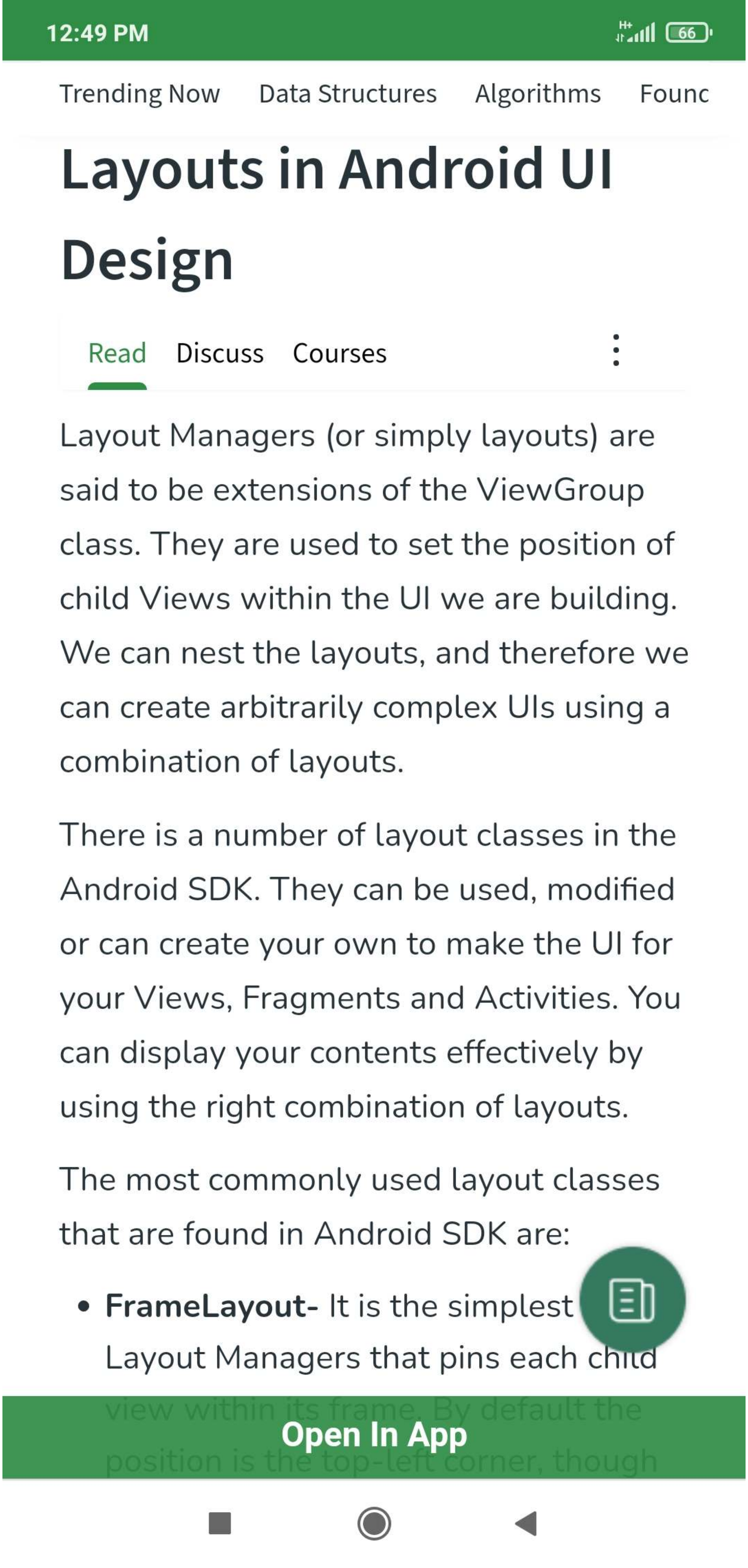
Step 2: Working with the XML Files

Android.

Step 2: Working with the XML Files

Next, go to the **activity_main.xml file**, which represents the UI of the project. Below is the code for the **activity_main.xml** file. Comments are added inside the code to understa e code in more detail.

XML



12:49 PM

Trending Now Data Structures Algorithms Founc

Layouts in Android UI Design

Read Discuss Courses

Layout Managers (or simply layouts) are said to be extensions of the ViewGroup class. They are used to set the position of child Views within the UI we are building. We can nest the layouts, and therefore we can create arbitrarily complex UIs using a combination of layouts.

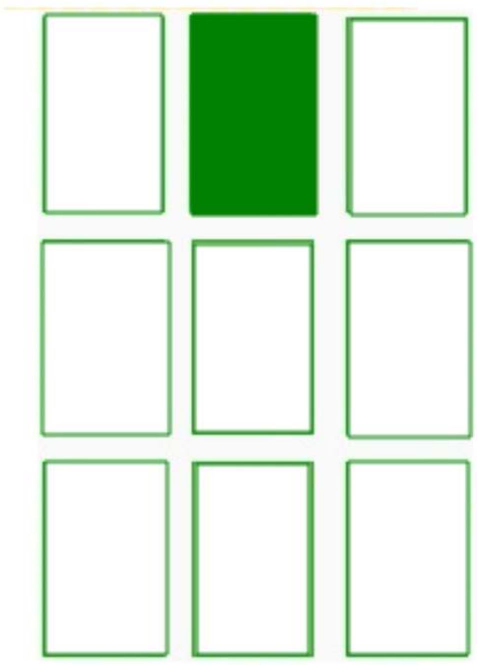
There is a number of layout classes in the Android SDK. They can be used, modified or can create your own to make the UI for your Views, Fragments and Activities. You can display your contents effectively by using the right combination of layouts.

The most commonly used layout classes that are found in Android SDK are:

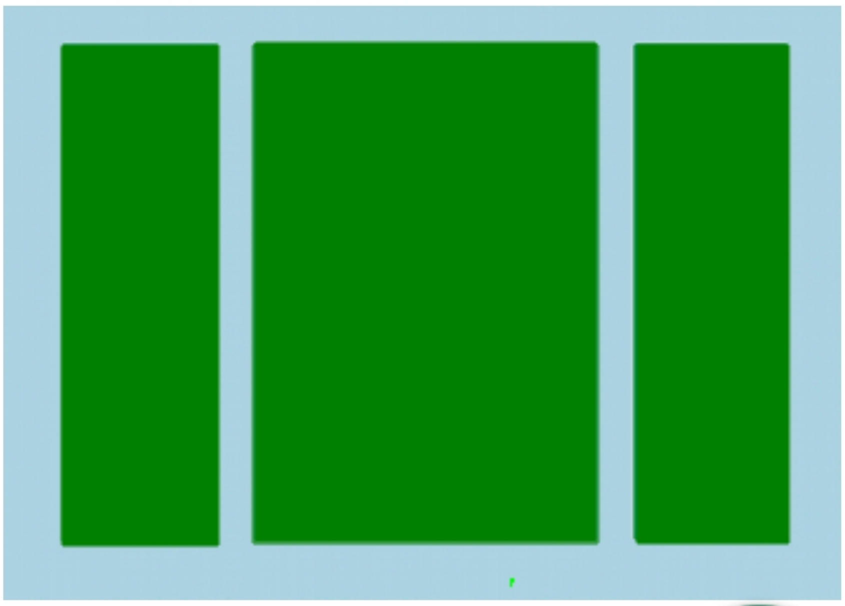
- **FrameLayout-** It is the simplest Layout Managers that pins each child view within its frame. By default the position is the top-left corner, though

Open In App

- **FrameLayout-** It is the simplest of the Layout Managers that pins each child view within its frame. By default the position is the top-left corner, though the gravity attribute can be used to alter its locations. You can add multiple children stacks each new child on top of the one before, with each new View potentially obscuring the previous ones.

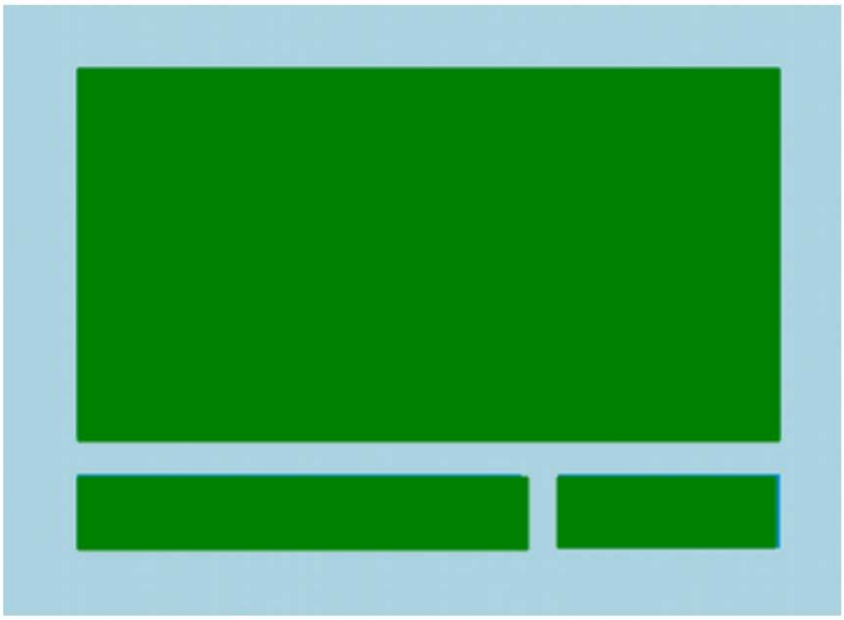


- **LinearLayout-** A LinearLayout aligns each of the child View in either a vertical or a horizontal line. A vertical layout has a column of Views, whereas in a horizontal layout there is a row of Views. It supports a weight attribute for each child View that can control the relative size of each child View within the available space.



- **RelativeLayout-** It is flexible than other native layouts as it lets us to define the position of each child View

- **RelativeLayout-** It is flexible than other native layouts as it lets us to define the position of each child View relative to the other views and the dimensions of the screen.



- **GridLayout-** It was introduced in Android 4.0 (API level 14), the Grid Layout used a rectangular grid of infinitely thin lines to lay out Vie series of rows and columns. The G Layout is incredibly flexible and can be used to greatly simplify layouts and

- **GridLayout-** It was introduced in Android 4.0 (API level 14), the Grid Layout used a rectangular grid of infinitely thin lines to lay out Views in a series of rows and columns. The Grid Layout is incredibly flexible and can be used to greatly simplify layouts and reduce or eliminate the complex nesting often required to construct UIs using the layouts described before.

Each of these layouts is designed to scale to suit the screen size of the host device by avoiding the used of absolute co-ordinates of the positions or predetermined pixel values. This makes the app suitable for the diverse set of Android devices.

Last Updated : 19 Sep, 2023

Similar Reads

Android UI Layouts

Open In App

Material Design Buttons in Android with Example

walks you through how to use Swing to create applications that feature key GUI components.

Most computer users today expect software to feature a graphical user interface (GUI) with a variety of widgets such as text boxes, sliders, and scrollbars. The Java Class Library includes Swing, a set of packages that enable Java programs to offer a sophisticated GUI and collect user input with the mouse, keyboard, and other input devices.

In this lesson, you will use Swing to create applications that feature these GUI components:

- **Frames:** Windows with a title bar; menu bar; and Maximize, Minimize, and Close buttons
- **Containers:** Components that hold other components
- **Buttons:** Clickable rectangles with text or graphics indicating their purpose
- **Labels:** Text or graphics that provide information
- **Text fields and text areas:** Windows that accept keyboard input of one line or multiple lines

- **Labels:** Text or graphics that provide information
- **Text fields and text areas:** Windows that accept keyboard input of one line or multiple lines
- **Drop-down lists:** Groups of related items that are selected from drop-down menus or scrolling windows
- **Check boxes and radio buttons:** Small squares or circles that can be selected or deselected
- **Image icons:** Graphics added to buttons, labels, and other components
- **Scrolling panes:** Panels for components too big for a user interface that can be accessed in full by using a scrollbar

Swing is the most extensive set of related classes introduced thus far in the book. Learning to create graphical applications with these packages is good practice for utilizing a class library in Java, which is something you'll do often in your own projects.

Creating an Application

Swing enables the creation of a Java program with an interface that adopts the style of the native operating system, such as Windows or Linux, or a style that's unique ing a class library in Java, which is something you'll do often in your own projects.

Creating an Application

Swing enables the creation of a Java program with an interface that adopts the style of the native operating system, such as Windows or Linux, or a style that's unique to Java. Each of these styles is called a *look and feel* because it describes both the appearance of the interface and how its components function when they are used.

Java offers a distinctive look and feel called Nimbus that's unique to the language.

Swing components are part of the `javax.swing` package, a standard part of the Java Class Library. To refer to a Swing class using its short name—without referring to the package, in other words—you must make it available with an `import` statement or use a catchall statement such as the following:

```
import javax.swing.*;
```

Two other packages that support GUI programming are `java.awt`, the Abstract Window Toolkit (AWT), and `java.awt.event`, event-handling classes that handle user input.

12:56 PM

```
import javax.swing.*;
```

Two other packages that support GUI programming are `java.awt`, the Abstract Window Toolkit (AWT), and `java.awt.event`, event-handling classes that handle user input.

When you use a Swing component, you work with objects of that component's class. You create the component by calling its constructor and then calling methods of the component as needed for proper setup.

All Swing components are subclasses of the abstract class `JComponent`. It includes methods to set a component's size, change the background color, define the font used for any displayed text, and set up *tooltips*. A tool-tip is explanatory text that appears when you hover the mouse over the component for a few seconds.

CAUTION

Swing classes inherit from many of the same super-classes as the Abstract Window Toolkit, so it is possible to use Swing and AWT components together in the same interface. However, the two types of com-ponents will not be rendered correctly in a container, Swing classes inherit from many of the same superclasses as the Abstract Window Toolkit, so it is possible to use Swing and AWT components together in the same interface. However, the two types of components will not be rendered correctly in a container, so it's best to always use Swing components; there's one for every AWT component.

Before components can be displayed in a user interface, they must be added to a *container*, a component that can hold other components. Swing containers are subclasses of `java.awt.Container`. This class includes methods to add and remove components from a container, arrange components using an object called a *layout manager*, and set up borders around the edges of a container. Containers often can be placed in other containers.

Creating a Graphical User Interface

The first step in creating a Swing application is to create a class that represents the main GUI. An object of this class serves as a container that holds all the other components to be displayed.

In many projects, the main interface object is a frame (the set up borders around the edges of a container. Containers often can be placed in other containers.

Creating a Graphical User Interface

The first step in creating a Swing application is to create a class that represents the main GUI. An object of this class serves as a container that holds all the other components to be displayed.

In many projects, the main interface object is a frame (the `JFrame` class in the `javax.swing` package). A frame is a window shown whenever you open an application on your computer. A frame has a title bar; Maximize, Minimize, and Close buttons; and other features.

In a graphical environment such as Windows or macOS, users expect to be able to move, resize, and close the windows of programs. One way to create a graphical Java application is to make the interface a subclass of `JFrame`, as in the following class declaration:

```
public class FeedReader extends JFrame {
    // body of class
}
```

The constructor of the class should handle the following tasks:

```
public class FeedReader extends JFrame {
    // body of class
}
```

The constructor of the class should handle the following tasks:

- Call a superclass constructor with `super()` to give the frame a title and handle other setup procedures.
- Set the size of the frame’s window, either by specifying the width and height in pixels or by letting Swing choose the size.
- Decide what to do if a user closes the window.
- Display the frame.

The `JFrame` class has the simple constructors `JFrame()` and `JFrame(`*`String`*`)`. One sets the frame’s title bar to the specified text, and the other leaves the title bar empty. You also can set the title by calling the frame’s `setTitle(`*`String`*`)` method.

The size of a frame can be established by calling the `setSize(`*`int`*`, `*`int`*`)` method with the width and height as arguments. A frame’s size is indicated in pixels, so, for

12:56 PM

- Display the frame.

The `JFrame` class has the simple constructors `JFrame()` and `JFrame(`*`String`*`)`. One sets the frame's title bar to the specified text, and the other leaves the title bar empty. You also can set the title by calling the frame's `setTitle(`*`String`*`)` method.

The size of a frame can be established by calling the `setSize(`*`int, int`*`)` method with the width and height as arguments. A frame's size is indicated in pixels, so, for example, calling `setSize(650, 550)` creates a frame 650 pixels wide and 550 pixels tall.

NOTE

You also can call the method `setSize(`*`Dimension`*`)` to set up a frame's size. `Dimension` is a class in the `java.awt` package that represents the width and height of a user interface component. Calling the `Dimension(`*`int, int`*`)` constructor creates a `Dimension` object representing the width and height specified as arguments.

Another way to set a frame's size is to fill the frame with the components it will contain and then call the frame's

12:56 PM

https://www.informit.com/articles

The `JFrame` class has the simple constructors `JFrame()` and `JFrame(`*`String`*`)`. One sets the frame's title bar to the specified text, and the other leaves the title bar empty. You also can set the title by calling the frame's `setTitle(`*`String`*`)` method.

The size of a frame can be established by calling the `setSize(`*`int`*`, `*`int`*`)` method with the width and height as arguments. A frame's size is indicated in pixels, so, for example, calling `setSize(650, 550)` creates a frame 650 pixels wide and 550 pixels tall.

NOTE

You also can call the method `setSize(`*`Dimension`*`)` to set up a frame's size. `Dimension` is a class in the `java.awt` package that represents the width and height of a user interface component. Calling the `Dimension(`*`int`*`, `*`int`*`)` constructor creates a `Dimension` object representing the width and height specified as arguments.

To change this, you must call a frame's `setDefaultCloseOperation(`*`int`*`)` method with one of four static variables as an argument:

- **`EXIT_ON_CLOSE`:** Exits the application when the frame is closed
- **`DISPOSE_ON_CLOSE`:** Closes the frame, removes the frame object from Java Virtual Machine (JVM) memory, and keeps running the application
- **`DO_NOTHING_ON_CLOSE`:** Keeps the frame open and continues running
- **`HIDE_ON_CLOSE`:** Closes the frame and continues running

These variables are part of the `JFrame` class because it implements the `WindowConstants` interface. To prevent a user from closing a frame, add the following statement to the frame's constructor method:

```
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_
```

If you are creating a frame to serve as an application's main user interface, the expected behavior is probably EXIT_ON_CLOSE which shuts down the application along tinues running

These variables are part of the JFrame class because it implements the WindowConstants interface. To prevent a user from closing a frame, add the following statement to the frame's constructor method:

```
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_
```

If you are creating a frame to serve as an application's main user interface, the expected behavior is probably EXIT_ON_CLOSE, which shuts down the application along with the frame.

As mentioned earlier, you can customize the overall appearance of a user interface in Java by designating a look and feel. The UIManager class in the javax.swing package manages this aspect of Swing. To set the look and feel, call the class method setLookAndFeel(*String*) with the name of the look and feel's class as the argument. Here's how to choose the Nimbus look and feel:

```
UIManager.setLookAndFeel(
    "javax.swing.plaf.nimbus.NimbusLookAndFeel
 );
```

This method call should be contained within a try-catch

This method call should be contained within a try-catch block because it might generate five different exceptions. Catching the Exception class and ignoring it causes the default look and feel to be used in the unlikely circumstance that Nimbus can't be chosen properly.

CAUTION

Using EXIT_ON_CLOSE shuts down the entire JVM, so it should be used only in the frame for an application's main window. If anything needs to happen after the frame closes, DISPOSE_ON_CLOSE or HIDE_ON_CLOSE should be used instead.

Developing a Framework

This lesson's first project is an application that displays a frame containing no other interface components. In NetBeans, create a new Java file with the class name SimpleFrame and the package name com.java21days and then enter Listing 9.1 as the source code. This simple application displays a frame 300×100 pixels in size and can serve as a framework—pun unavoidable—for any applications you create that use a GUI.

12:57 PM

Creating a Component

Creating a GUI is a great way to get experience working with objects in Java because each interface component is represented by its own class.

To use an interface component in Java, you create an object of that component's class. You already have worked with the container class `JFrame`.

One of the simplest components to employ is `JButton`, the class that represents clickable buttons.

In any program, buttons trigger actions. You could click Install to begin installing software, click a Run button to begin a new game of Angry Birds, click the Minimize button to prevent your boss from seeing Angry Birds running, and so on.

A Swing button can feature a text label, a graphical icon, or both.

You can use the following constructors for buttons:

- **`JButton(`*`String`*`)`:** A button labeled with the specified text
- **`JButton(`*`Icon`*`)`:** A button that displays the specified graphical icon
- **`JButton(`*`String, Icon`*`)`:** A button with the spe-

- **`JButton(`*`String`*`)`:** A button labeled with the specified text
- **`JButton(`*`Icon`*`)`:** A button that displays the specified graphical icon
- **`JButton(`*`String, Icon`*`)`:** A button with the specified text and graphical icon

The following statements create three buttons with text labels:

```
JButton play = new JButton("Play");
JButton stop = new JButton("Stop");
JButton rewind = new JButton("Rewind");
```

Graphical buttons with icons are covered later in this lesson.

Adding Components to a Container

Before you can display a user interface component such as a button in a Java program, you must add it to a container and display that container.

To add a component to a container, call the container's `add(`*`Component`*`)` method with the component as the argument. (All user interface components in Swing inherit from `java.awt.Component`.)

The following statements create three buttons with text labels:

```
JButton play = new JButton("Play");
JButton stop = new JButton("Stop");
JButton rewind = new JButton("Rewind");
```

Graphical buttons with icons are covered later in this lesson.

Adding Components to a Container

Before you can display a user interface component such as a button in a Java program, you must add it to a container and display that container.

To add a component to a container, call the container's `add(`*`Component`*`)` method with the component as the argument. (All user interface components in Swing inherit from `java.awt.Component`.)

The simplest Swing container is a panel (the `JPanel` class). The following example creates a button and adds it to a panel:

```
JButton quit = new JButton("Quit");
JPanel panel = new JPanel();
panel.add(quit);
```