**Services in Android**

**Android service** is a component that is *used to perform operations on the background* such as playing music, handle network transactions, interacting content providers etc. It doesn't has any UI (user interface).

The service runs in the background indefinitely even if application is destroyed.

Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).

The android.app.Service is subclass of ContextWrapper class.

Note: Android service is not a thread or separate process.

Life Cycle of Android Service

There can be two forms of a service.The lifecycle of service can follow two different paths: started or bound.

1. Started
2. Bound

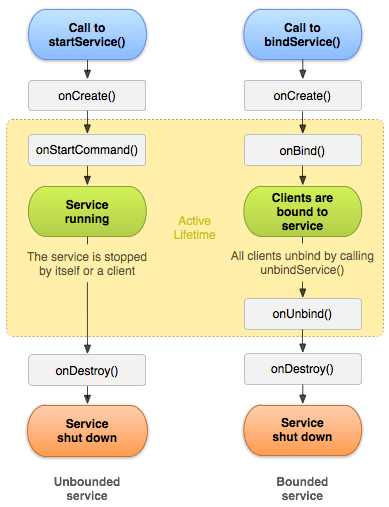
1) Started Service

A service is started when component (like activity) calls **startService()** method, now it runs in the background indefinitely. It is stopped by **stopService()** method. The service can stop itself by calling the **stopSelf()** method.

2) Bound Service

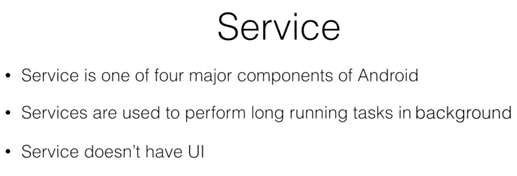
A service is bound when another component (e.g. client) calls **bindService()** method. The client can unbind the service by calling the **unbindService()** method.

The service cannot be stopped until all clients unbind the service.



Understanding Started and Bound Service by background music example

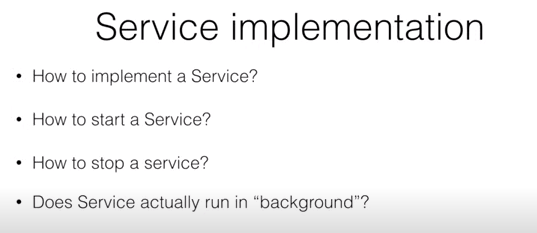
Suppose, I want to play music in the background, so call startService() method. But I want to get information of the current song being played, I will bind the service that provides information about the current song.



Android app components are Activity ,Services ,Broadcast Receivers ,Content Provider.

Long running tasks in background means like downloading a file .

Services do not have UI because we don’t want to keep the user busy by showing download progress continuously that’s why services run without UI



Refer the below code for the implementation

MainActivity.java

**public class** MainActivity **extends** AppCompatActivity {  
 Button **start\_btn**,**stop\_btn**;  
 Intent **service**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **start\_btn**=findViewById(R.id.***start\_bt***);  
 **stop\_btn**=findViewById(R.id.***stop\_bt***);  
 **service**=**new** Intent(**this**,MyService.**class**);  
 Toast.*makeText*(**this**,**"Main Thread is"**+Thread.*currentThread*().getId(),Toast.***LENGTH\_LONG***).show();  
 **start\_btn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 startService(**service**);  
 }  
 });  
 }  
}

MyService.java

**public class** MyService **extends** Service {  
  
 @Override  
 **public int** onStartCommand(Intent intent, **int** flags, **int** startId) {  
 Log.*i*(**"Harry"**,**"Service thread is"**+Thread.*currentThread*().getId());  
 *// stopSelf(); without it we can not stop a service* **return super**.onStartCommand(intent, flags, startId);  
 }  
  
 @Override  
 **public void** onDestroy() {  
 Log.*i*(**"Harry"**,**"Service end "**+Thread.*currentThread*().getId());  
 **super**.onDestroy();  
 }  
  
 @Override  
 **public** IBinder onBind(Intent intent) { // this will come into picture when we will discuss Bound services.  
 **return null**;  
 }  
}

Manifest File

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.service\_basic\_example"**>  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 <**service android:name=".MyService"**/>  
 </**application**>  
  
</**manifest**>

In the above example Service is started by an Activity. Service is running in background but running on same UI thread. It is not using separate thread here.

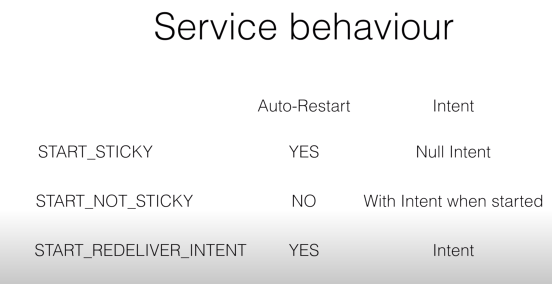
**Behavior of Service**

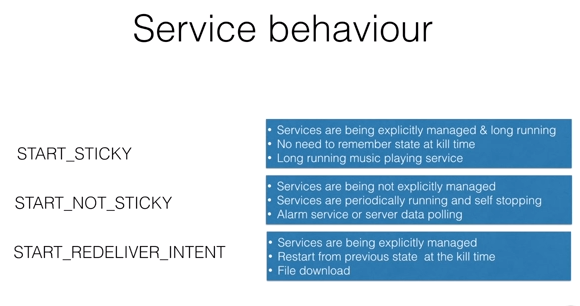
We have seen onStartCommand method returns a integer value. To Understand this, let us see what happens with app when resource crunch situation occur in android.

For example I have opened many apps in my phone. So it may arise the situation of resource crunch and then android may need to decide to kill my app that is not running in the foreground. In this case my app is running a service in the background then android will not try to kill that particular app because service always get a higher priority.but that doesnot mean you keep a service u will not get killed.if resource crunch is seviour then android may kill that app with service too.

To understand the service behavior

When resource crunch happened,service app killed by os . in this case what should happen to killed service? That is determined by the integer value returned by the onstartCommand method.

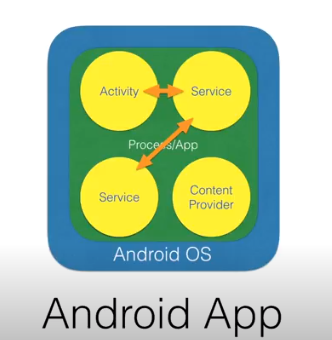




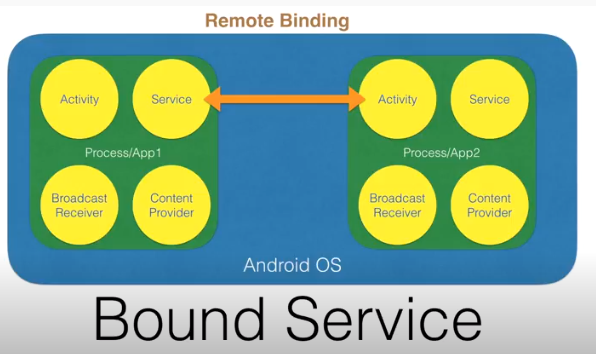
**Bound Services**

In Android, An app is treated as a separate process. An app can have these four components as discussed previous Activity, Service, Broadcast Receiver and Content Provider.

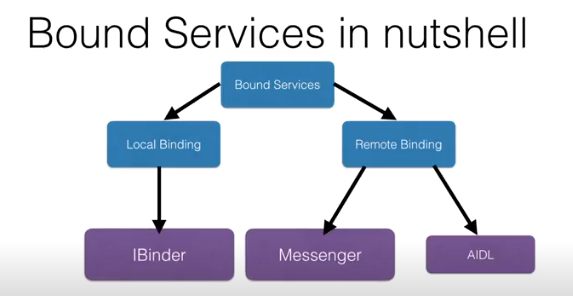
An Activity may bind to a service to get some status update. It need not to be every time the activity needed , any service can be bind with another service to get some status update. In either to these case,Activity is trying to bind to a service and the service which is providing the information to the activity or service is known as **Bound Service.**



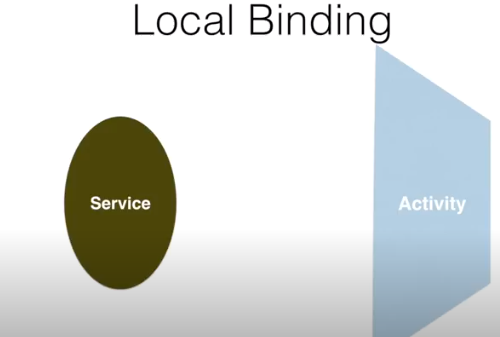
We have seen in above diagram, the components which are interested in establishing the connection to the service are part of the same app and that is why we can call it as **Local Binding.**

****

When the Activity of one app is trying to bind with a service who belongs to another app. Then This is known as Remote Binding as displayed in above diagram.

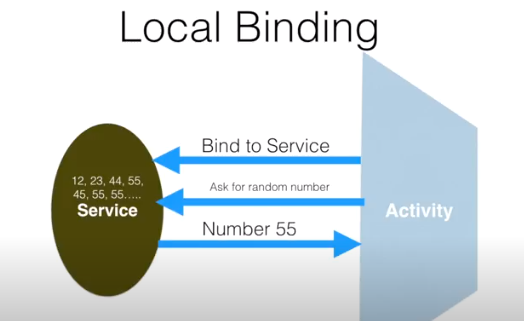


* Local Binding is implemented by IBinder Interface.
* For Remote Binding can be implemented using two mechanism, we use Messenger Api and AIDL (Android Interface Definition Langauage)
* Messenger is basically a queued concept where in any component which want to connect to service will trigger a request and these are queued. This is most suited for Non Multi threaded scenario.
* AIDL is more complex and highly suitable for Multithreaded environment.

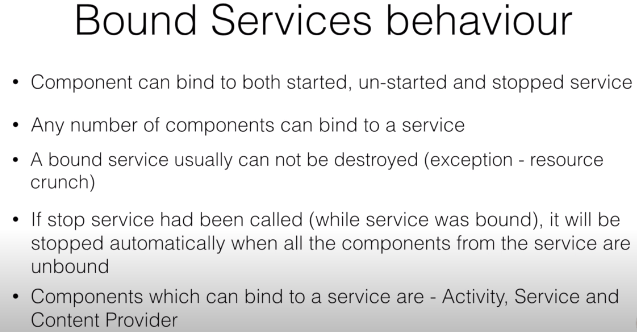


Activity is a visible part and service is invisible part works In background.

Lets take an example,In Background Service is continuously generating Random Numbers. Firstly Activity will bind to the service. If binding is successful, it will ask for random number. In response service will return a random number to Activity.

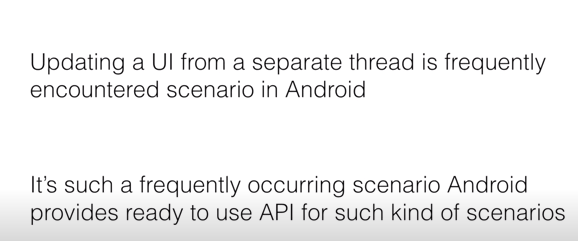


* To Achieve this **Service** need to implement onBind method which will return IBinder object.
* The Activity will use the ServiceConnection Api to connect with service.

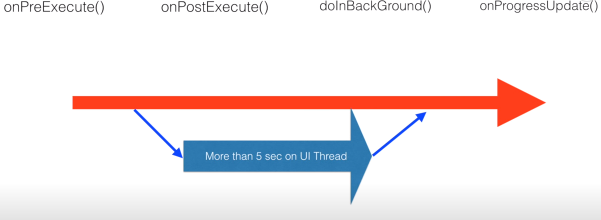


That’s how we can bound the activity with a service.





That Api Is known as Asynctask.



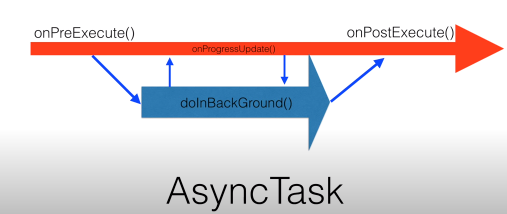
These four methods basically make sure that you are able to do long task in a way you don’t disturb the main thread.

**(Params)** You use **doinBackground()** to perform long task on the separate thread.Android automatically run anything or everything that you code in doinBackground() on a separate thread.

**onPreExecute()** this part gets executed on the main thread .you typically do initialization bfore you want to do anything in the background.

**(Progress…) onPostExecute()** once again execute on the UI thread. Whatever you have done ,you want to update on your Ui.and the best way to do is on UI thread.

One more scenario where in intermediate you want to update the Ui in between execution of doinBackground() or we can say to show the progress even you are doing something in background , we use **onProgressUpdate() (Result)**

****

## Generic Types in Async Task

* **TypeOfVarArgParams** − It contains information about what type of params used for execution.
* **ProgressValue** − It contains information about progress units. While doing background operation we can update information on ui using onProgressUpdate().
* **ResultValue** −It contains information about result type.

**Combine example of Asynctask ,Threading and JSON Parsing**

Xml file

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity"**>  
  
 <**ScrollView  
 android:layout\_weight="1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"**>  
 <**TextView  
 android:textSize="20sp"  
 android:text="CLICK"  
 android:id="@+id/tv\_result"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />  
 </**ScrollView**>  
 <**ProgressBar  
 android:id="@+id/progress"  
 android:layout\_gravity="center"  
 style="@style/Widget.AppCompat.ProgressBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"** />  
  
 <**Button  
 android:text="Show"  
 android:layout\_weight="0.2"  
 android:id="@+id/bt\_show"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"** />  
  
</**LinearLayout**>

Java file

**package** com.example.asynctask\_thread\_example;  
  
**import** android.os.AsyncTask;  
**import** android.os.Handler;  
**import** android.os.Looper;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.ProgressBar;  
**import** android.widget.TextView;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.net.HttpURLConnection;  
**import** java.net.MalformedURLException;  
**import** java.net.URL;  
  
  
**public class** MainActivity **extends** AppCompatActivity {  
 String **url** = **"http://api.plos.org/search?q=title:%22Drosophila%22%20and%20body:%22RNA%22&fl=id,abstract"**;  
 String **current** = **""**;  
 **public final** String **TAG** = **"Harry"**;  
 Runnable **runnable**;  
 ProgressBar **bar**;  
 TextView **tv**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **tv** = findViewById(R.id.***tv\_result***);  
 Button b = findViewById(R.id.***bt\_show***);  
 **bar** = findViewById(R.id.***progress***);  
 **bar**.setVisibility(View.***INVISIBLE***);  
 b.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **bar**.setVisibility(View.***VISIBLE***);  
  
 *// try {  
 // Thread.sleep(2000);  
 // } catch (InterruptedException e) {  
 // e.printStackTrace();  
 // }  
 // Thread tt=new Thread(runnable);  
 // tt.start();* **new** Asyn().execute(**url**);  
  
 }  
 });  
  
 **runnable** = **new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
 URL web = **new** URL(**url**);  
 HttpURLConnection con = (HttpURLConnection) web.openConnection();  
 InputStream in = con.getInputStream();  
 InputStreamReader isw = **new** InputStreamReader(in);  
 **int** data = isw.read();  
 **while** (data != -1) {  
 **current** += (**char**) data;  
 data = isw.read();  
 }  
 *// tv.append(current);* Log.*v*(**TAG**, **current**);  
 } **catch** (MalformedURLException e1) {  
 e1.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 };  
  
  
 }  
  
 **public class** Asyn **extends** AsyncTask<String, Void, String> { *//<param,progress,result> datatyps* @Override  
 **protected** String doInBackground(String... strings) {  
 **try** {  
 URL web = **new** URL(strings[0]);  
 HttpURLConnection con = (HttpURLConnection) web.openConnection();  
 InputStream in = con.getInputStream();  
 InputStreamReader isw = **new** InputStreamReader(in);  
 **int** data = isw.read();  
 **while** (data != -1) {  
 **current** += (**char**) data;  
 data = isw.read();  
 }  
 *// tv.append(current.toString());* Log.*v*(**TAG**, **current**);  
 } **catch** (MalformedURLException e1) {  
 e1.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return current**;  
 }  
  
  
 **protected void** onPostExecute(String data) {  
 **bar**.setVisibility(View.***GONE***);  
 **tv**.append(data);  
*// try {  
 // parseData(data);  
// } catch (JSONException e) {  
// e.printStackTrace();  
// }* }  
 }  
  
 **public void** parseData(String data) {  
  
 JSONObject root = **null**;  
 **try** {  
 root = **new** JSONObject(data);  
  
 JSONObject response = root.getJSONObject(**"response"**);  
 String numFound = response.getString(**"numFound"**);  
 **int** start = response.getInt(**"start"**);  
 Log.*i*(**TAG**, numFound);  
 Log.*i*(**TAG**, start + **""**);  
 JSONArray docs = response.getJSONArray(**"docs"**);  
 **for** (**int** i = 0; i < docs.length(); i++) {  
 JSONObject arraydata = docs.getJSONObject(i);  
 String id = arraydata.getString(**"id"**);  
 Log.*i*(**TAG**, id);  
 JSONArray abstr = arraydata.getJSONArray(**"abstract"**);  
 **for** (**int** j = 0; j < abstr.length(); j++) {  
 *//JSONObject abs = abstr.getJSONObject(j);* String abst = abstr.getString(j);  
 Log.*i*(**TAG**, abst);  
  
 }  
 }  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
}