[Android Mobile Application Development](http://upadhyayjiteshandroid.blogspot.in/)

Tuesday, October 15, 2013

**Difference between bound and unbound service in Android**

*Android Bound Service and Unbound Service run in Background Android Services is used to do any background task in current activity.There is two types of Services available in android.Service class is responsible for using Services in android for that we need to extends Services class in out Custom service.*

***Bound Service****:Service which call indefinitely in between activity*

*An Android component may bind itself to a Service using bindservice (). A bound service would run as long as the other application components are bound to it. As soon as they unbind, the service destroys itself.*

***Unbound Service****:Service which call at the life span of calling activity*

*In this case, an application component starts the service, and it would continue to run in the background, even if the original component that initiated it is destroyed. For instance, when started, a service would continue to play music in the background indefinitely.*

*Services has different life cycle from Activity. Services has different method for their life cycle which is startService(),stopService(),onBindService()*

[Can anybody explain what is difference between unbound and bound service in android](http://stackoverflow.com/questions/25240299/can-anybody-explain-what-is-difference-between-unbound-and-bound-service-in-andr)

[](http://engine.adzerk.net/r?e=&s=FKKvFavdDLMzX4M5VJV7Kkkbmzo)

|  |  |  |  |
| --- | --- | --- | --- |
| up vote8down vote[favorite](http://stackoverflow.com/questions/25240299/can-anybody-explain-what-is-difference-between-unbound-and-bound-service-in-andr)  3 | Can anybody explain what is difference between unbound and bound service in android and explain about intent service  Thanks  [[http://i.stack.imgur.com/tKsDb.png](http://stackoverflow.com/questions/tagged/android)android](http://stackoverflow.com/questions/tagged/android)   |  |  | | --- | --- | | [share](http://stackoverflow.com/q/25240299)[improve this question](http://stackoverflow.com/posts/25240299/edit) | asked Aug 11 '14 at 9:43  [[https://www.gravatar.com/avatar/26e1dc3350c5320680a866d719ead78f?s=32&d=identicon&r=PG](http://stackoverflow.com/users/426344/mohan)](http://stackoverflow.com/users/426344/mohan)  [mohan](http://stackoverflow.com/users/426344/mohan)  **3,125**1772146 | |
|  | add a comment |

2 Answers

[active](http://stackoverflow.com/questions/25240299/can-anybody-explain-what-is-difference-between-unbound-and-bound-service-in-andr?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/25240299/can-anybody-explain-what-is-difference-between-unbound-and-bound-service-in-andr?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/25240299/can-anybody-explain-what-is-difference-between-unbound-and-bound-service-in-andr?answertab=votes#tab-top)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote14down voteaccepted | [**Bounded Service**](http://developer.android.com/guide/components/bound-services.html)  A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).  When the last client unbinds from the service, the system destroys the service **EXCEPT** If the service was started by [startService](http://developer.android.com/reference/android/content/Context.html" \l "startService(android.content.Intent))  **Unbounded Service or Started**  A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.  **BUT**  Most confusion about the Service class actually revolves around what it is not:  A Service is not a separate process. The Service object itself does not imply it is running in its own process; unless otherwise specified, it runs in the same process as the application it is part of.  A Service is not a thread. It is not a means itself to do work off of the main thread (to avoid Application Not Responding errors).  That is where **IntentService** are used.  [**IntentService**](http://developer.android.com/guide/components/services.html#ExtendingIntentService) is a subclass of Service that uses a worker thread to  handle all start asynchronous requests (expressed as Intents) on demand, one at a time. Clients send requests through startService(Intent) calls; the service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work.  Example  hope it helps :)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/25240537)[improve this answer](http://stackoverflow.com/posts/25240537/edit) | [edited Dec 17 '15 at 15:56](http://stackoverflow.com/posts/25240537/revisions) | answered Aug 11 '14 at 9:56  [[https://i.stack.imgur.com/ifXu0.png?s=32&g=1](http://stackoverflow.com/users/2778376/spurdow)](http://stackoverflow.com/users/2778376/spurdow)  [Spurdow](http://stackoverflow.com/users/2778376/spurdow)  **1,136**518 | |

## [How to create simple (unbound) Service in Android?](https://priyankacool10.wordpress.com/2014/04/27/how-to-create-simple-unbound-service-in-android/)

Posted on [April 27, 2014](https://priyankacool10.wordpress.com/2014/04/27/how-to-create-simple-unbound-service-in-android/)by [Priyanka Kapoor](https://priyankacool10.wordpress.com/author/priyankacool/" \o "View all posts by Priyanka Kapoor)

## ****Services in Android System****

**1) Faceless components:** A service is a component which runs in the background without direct interaction with the user. As the service has no user interface, it is not bound to the lifecycle of an activity.

**2) Taking care of long running background tasks:** Services are used for repetitive and potentially long running operations, i.e., Internet downloads, checking for new data, data processing, updating content providers and the like.

**3) Runs on UI Thread:** By default a service runs in the same process as the main thread of the application.

Therefore you need to use asynchronous processing in the service to perform resource intensive tasks in the background. A common used pattern for a service implementation is to create and run a new Thread in the service to perform the processing in the background and then to terminate the service once it has finished the processing.

Services which run in the process of the application are sometimes called local services.

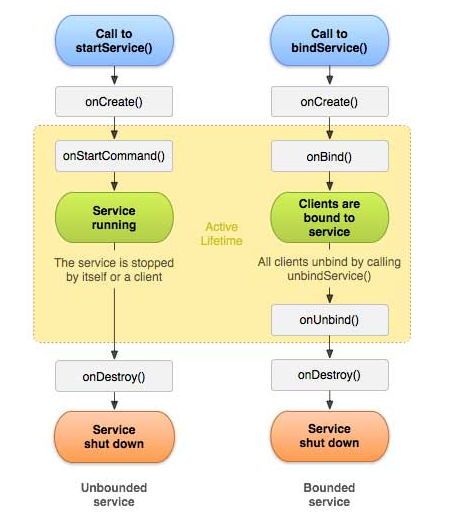
**4)Unaffected by activity switching:**Each Service has a specific job, and they keep at it if you switch between different Activities, or even if you switch to a different application altogether.

In Android a Service is used for two main reason:

* Implement multi-task
* Enable Inter-Process-Communication (IPC)

A typical example of the first case is an app that required to download data from a remote server, in this case we can have Activity that interacts with a user and starts a service that accomplishes the work in background while the user can use the app and maybe when the service finishes sends a message to the user.

In the second case, we want to “share” some common functions so that different app can re-use them. For example, we can suppose we have a service that sends an email and we want to share this service among several apps so that we don’t have to rewrite the same code. In this case we can use IPC so that the service exposes a “remote” interface that can be called by other app.

[](https://priyankacool10.files.wordpress.com/2014/04/service_lifecycle.png)

## ****Forms of Service****

A Service can have two forms:

**1) Started/Unbound:** In this case, an application component starts the service by calling *startService()* , and it would **continue to run in the background, even if the original component that initiated it is destroyed**. For instance, when started, a service would continue to play music in the background indefinitely.

**2) Bound:** An Android component may bind itself to a Service using bindservice (). A bound service would **run as long as the other application components are bound to it**. As soon as they unbind, the service destroys itself.

An unbound activity runs indefinitely, whereas the lifespan of a bound activity depends on the application components that bind to it.

The following diagram on the left shows the lifecycle when the service is created with startService() and the diagram on the right shows the lifecycle when the service is created with bindService(): *(image courtesy : android.com )*

## CallBacks of Service:

|  |  |
| --- | --- |
| **Callback** | **Description** |
| onStartCommand() | The system calls this method when another component, such as an activity, requests that the service be started, by calling *startService()*. If you implement this method, it is your responsibility to stop the service when its work is done, by calling *stopSelf()* or *stopService()*methods. |
| onBind() | The system calls this method when another component wants to bind with the service by calling *bindService()*. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don’t want to allow binding, then you should return *null*. |
| onUnbind() | The system calls this method when all clients have disconnected from a particular interface published by the service. |
| onRebind() | The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*. |
| onCreate() | The system calls this method when the service is first created using*onStartCommand()* or *onBind()*. This call is required to perform one-time setup. |
| onDestroy() | The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc. |

onStartCommand() method has integer return type value which can be any of the following:

START\_STICKY : Using this return value, if the OS kills our Service it will recreate it but the Intent that was sent to the Service isn’t redelivered. In this way the Service is always running

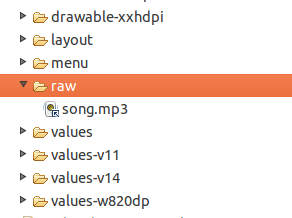
START\_NOT\_STICKY: If the OS kills the Service it won’t recreate it until the client calls explicitly onStart command

START\_REDELIVER\_INTENT: It is similar to the START\_STICKY and in this case the Intent will be redelivered to the service.

## Steps to Create a Simple Service:

This code starts and stops a simple service, plays a music in background.

1) Firstly create “raw” folder inside res folder and place music file in it. Here it is song.mp3

[](https://priyankacool10.files.wordpress.com/2014/04/service1.png)

2) Add the service in manifest file:

<service

android:name=".MyService"/>

3) Layout will contain two buttons play and stop. Play will start the service of playing music and Stop will stop the music.

(layout\_main.xml)

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/play"/>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/stop"/>

</LinearLayout>

The MainActivity.java file calls the service on pressing the button:

package com.example.simpleservice;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.widget.Button;

import android.view.View;

import android.view.View.OnClickListener;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.layout\_main);

Button play,stop;

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

play.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View view){

Intent myIntent = new Intent(MainActivity.this, MyService.class);

startService(myIntent);

}

});

stop=(Button)findViewById(R.id.button2);

stop.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View view){

Intent myIntent = new Intent(MainActivity.this, MyService.class);

stopService(myIntent);

}

});

}

}

The main Service code resides in MyService.java file:

package com.example.simpleservice;

import android.app.Service;

import android.content.Intent;

import android.media.MediaPlayer;

import android.os.IBinder;

public class MyService extends Service{

String tag="MyService";

MediaPlayer mp;

@Override

public IBinder onBind(Intent intent){

return null;

}

@Override

public void onCreate(){

super.onCreate();

mp = MediaPlayer.create(getApplicationContext(), R.raw.song);

}

@Override

public int onStartCommand(Intent intent, int flags, int startId){

mp.start();

return START\_STICKY;

}

@Override

public void onDestroy(){

mp.release();

super.onDestroy();

}

}

Its done. Click on play button to play the music and stop button to stop the music.

About these ads