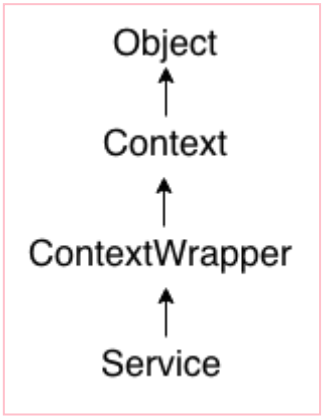
Android Service Tutorial

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

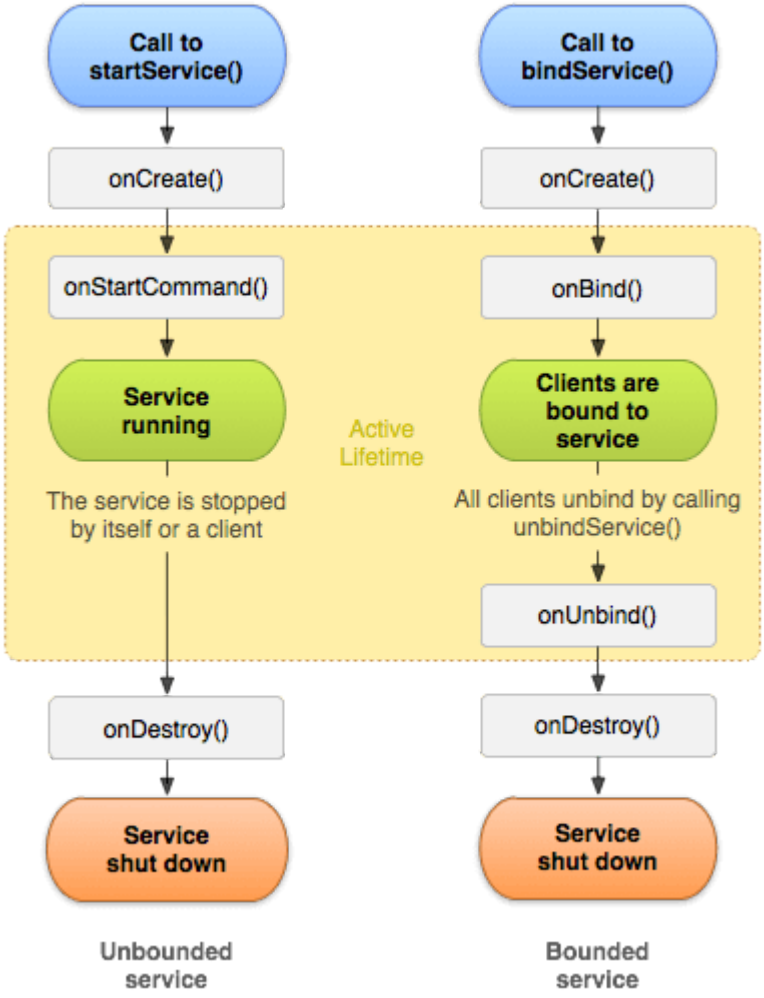
1) Started Service

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.

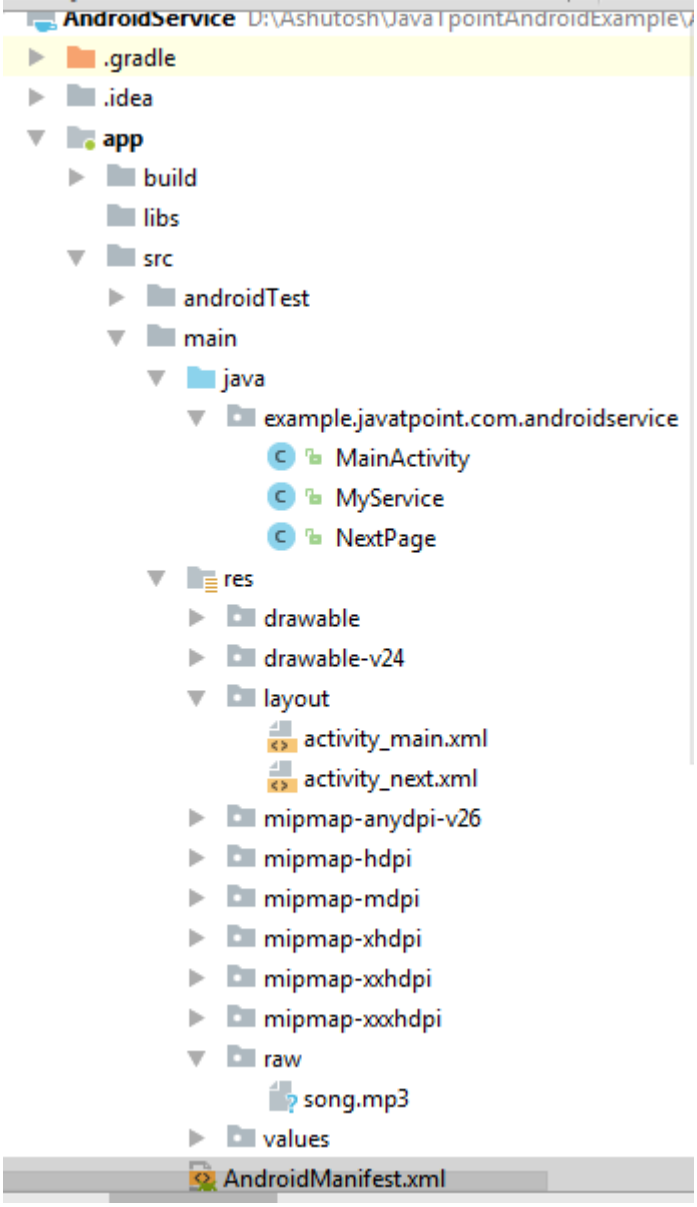


Understanding Started and Bound Service by background music example

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 Service Example

Let's see the example of service in android that plays an audio in the background. Audio will not be stopped even if you switch to another activity. To stop the audio, you need to stop the service.



activity_main.xml

Drag the 3 buttons from the pallete, now the activity_main.xml will look like this:

File: activity_main.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="example.javatpoint.com.androidservice.MainActivity">



    <Button
        android:id="@+id/buttonStart"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="74dp"
    android:text="Start Service" />

  <Button
    android:id="@+id/buttonStop"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="Stop Service" />

  <Button
    android:id="@+id/buttonNext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="63dp"
    android:text="Next Page" />
</RelativeLayout>
```

---

activity_next.xml

It is the layout file of next activity.

File: activity_next.xml

It contains only one textview displaying the message Next Page

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/a
  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="example.javatpoint.com.androidservice.NextPage">
```

```
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="200dp"
        android:text="Next Page"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Service class

Now create the service implemenation class by inheriting the Service class and overridding its callback methods.

File: MyService.java

```
package example.javatpoint.com.androidservice;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.widget.Toast;

public class MyService extends Service {
    MediaPlayer myPlayer;
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        Toast.makeText(this, "Service Created", Toast.LENGTH_LONG).show();
```

```
        myPlayer = MediaPlayer.create(this, R.raw.sun);
        myPlayer.setLooping(false); // Set looping
    }
    @Override
    public void onStart(Intent intent, int startid) {
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        myPlayer.start();
    }
    @Override
    public void onDestroy() {
        Toast.makeText(this, "Service Stopped", Toast.LENGTH_LONG).show();
        myPlayer.stop();
    }
}
```

Activity class

Now create the MainActivity class to perform event handling. Here, we are writing the code to start and stop service. Additionally, calling the second activity on buttonNext.

File: MainActivity.java

```
package example.javatpoint.com.androidservice;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    Button buttonStart, buttonStop,buttonNext;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        buttonStart = findViewById(R.id.buttonStart);
        buttonStop = findViewById(R.id.buttonStop);
```

```
        buttonNext =  findViewById(R.id.buttonNext);

        buttonStart.setOnClickListener(this);
        buttonStop.setOnClickListener(this);
        buttonNext.setOnClickListener(this);



    }
    public void onClick(View src) {
        switch (src.getId()) {
            case R.id.buttonStart:

                startService(new Intent(this, MyService.class));
                break;
            case R.id.buttonStop:
                stopService(new Intent(this, MyService.class));
                break;
            case R.id.buttonNext:
                Intent intent=new Intent(this,NextPage.class);
                startActivity(intent);
                break;
        }
    }
}
```

NextPage class

Now, create another activity.

File: NextPage.java

```
package example.javatpoint.com.androidservice;

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

public class NextPage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
    }
}
```

Declare the Service in the AndroidManifest.xml file

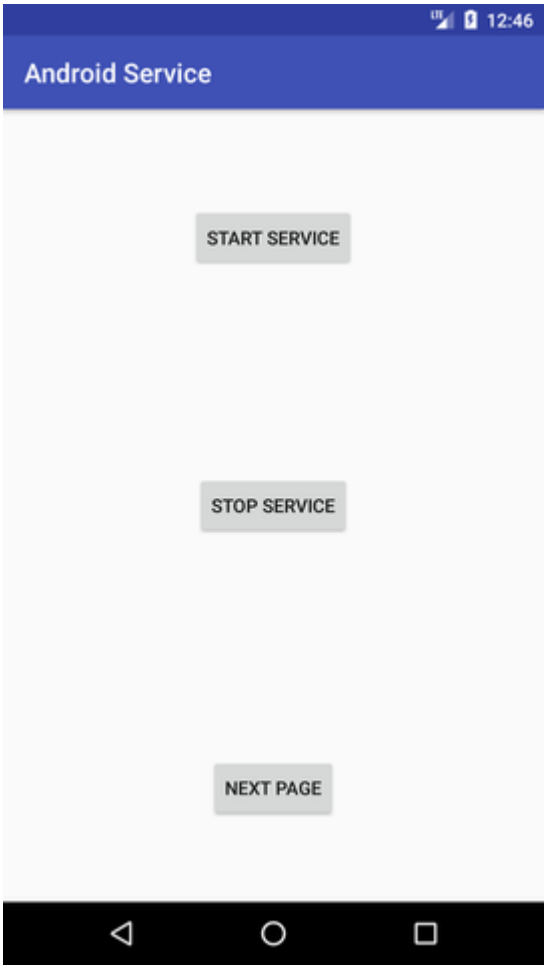
Finally, declare the service in the manifest file.

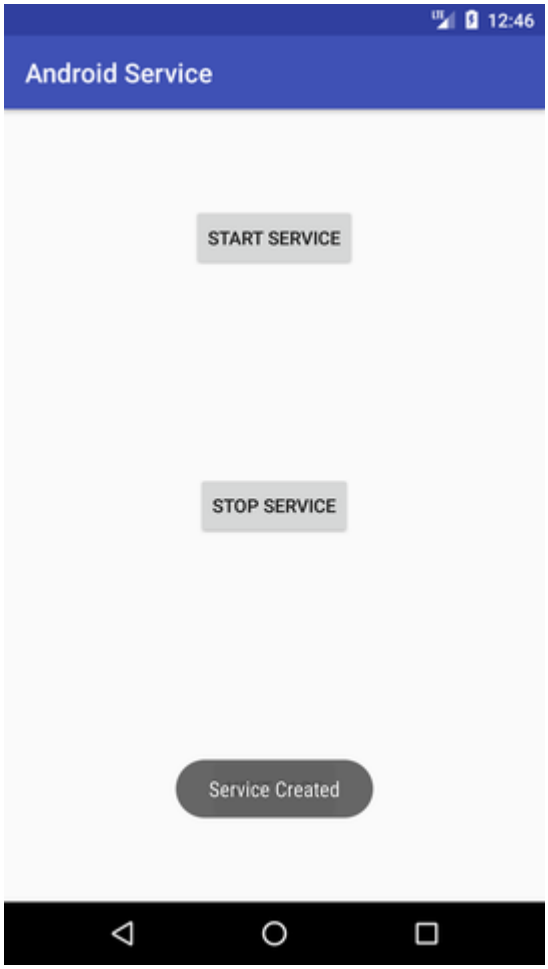
File: AndroidManifest.xml

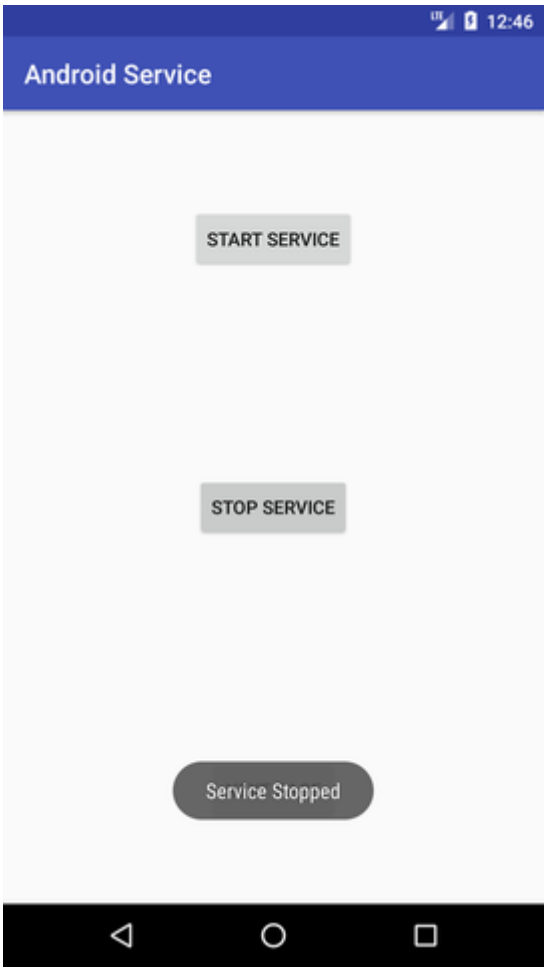
Let's see the complete AndroidManifest.xml file

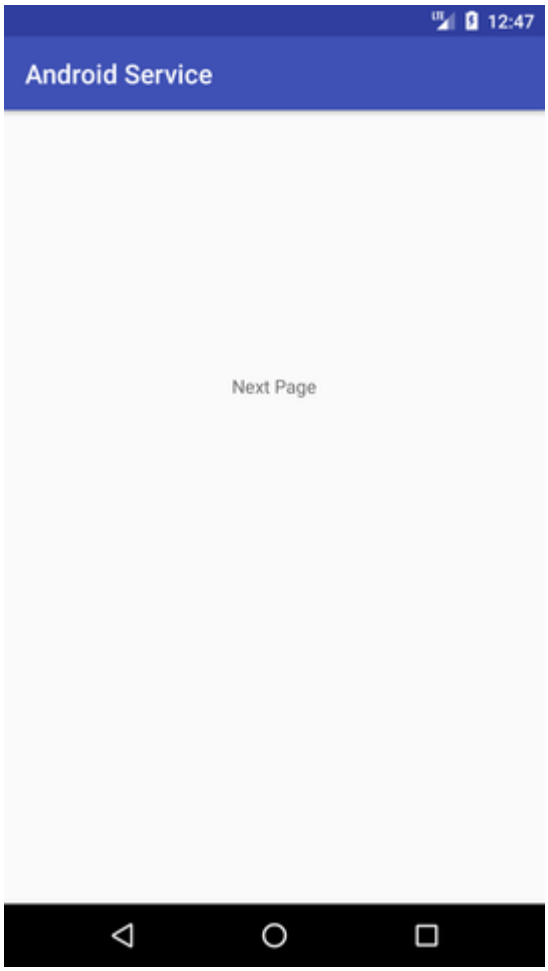
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.javatpoint.com.androidservice">

    <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>
        <activity android:name=".NextPage"></activity>
        <service
            android:name=".MyService"
            android:enabled="true" />
    </application>

</manifest>
```

Output:









← Prev

Next →

For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

R Programming

RxJS

React Native

Python Design Patterns

Python Pillow

Python Turtle

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence Tutorial

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing tutorial

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System tutorial

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking Tutorial

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering Tutorial

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse