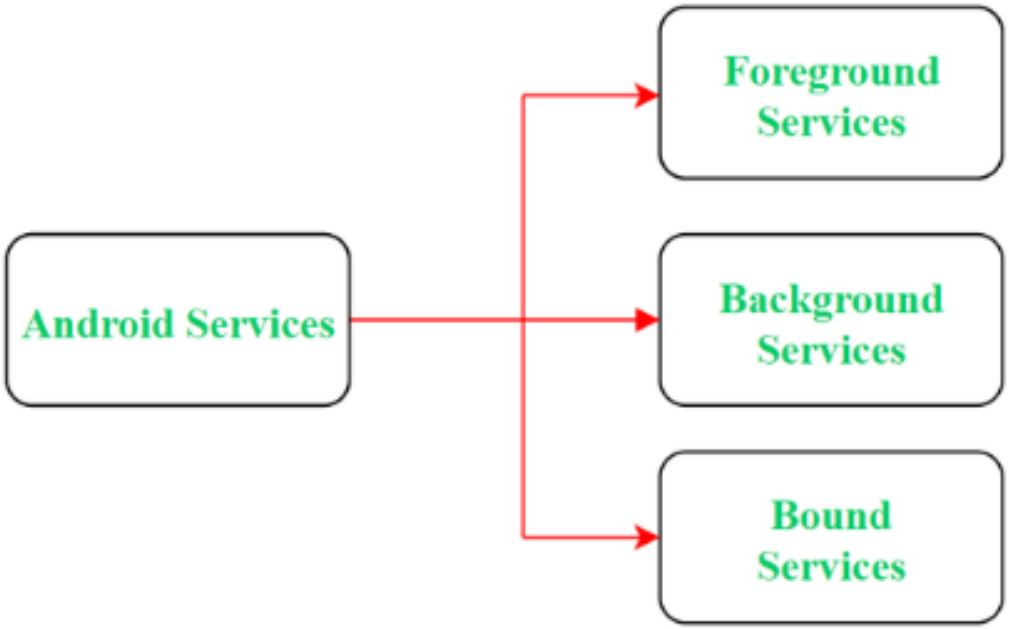
Services

What is service?

- A Service is an application component that can perform long-running operations in the background.
- It does not provide a user interface.
- Once started, a service might continue running for some time, even after the user switches to another application.
- Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC).
- For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Types of Services



Types of Services

Foreground

A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a Notification. Foreground services continue running even when the user isn't interacting with the app.

When you use a foreground service, you must display a notification so that users are actively aware that the service is running. This notification cannot be dismissed unless the service is either stopped or removed from the foreground.

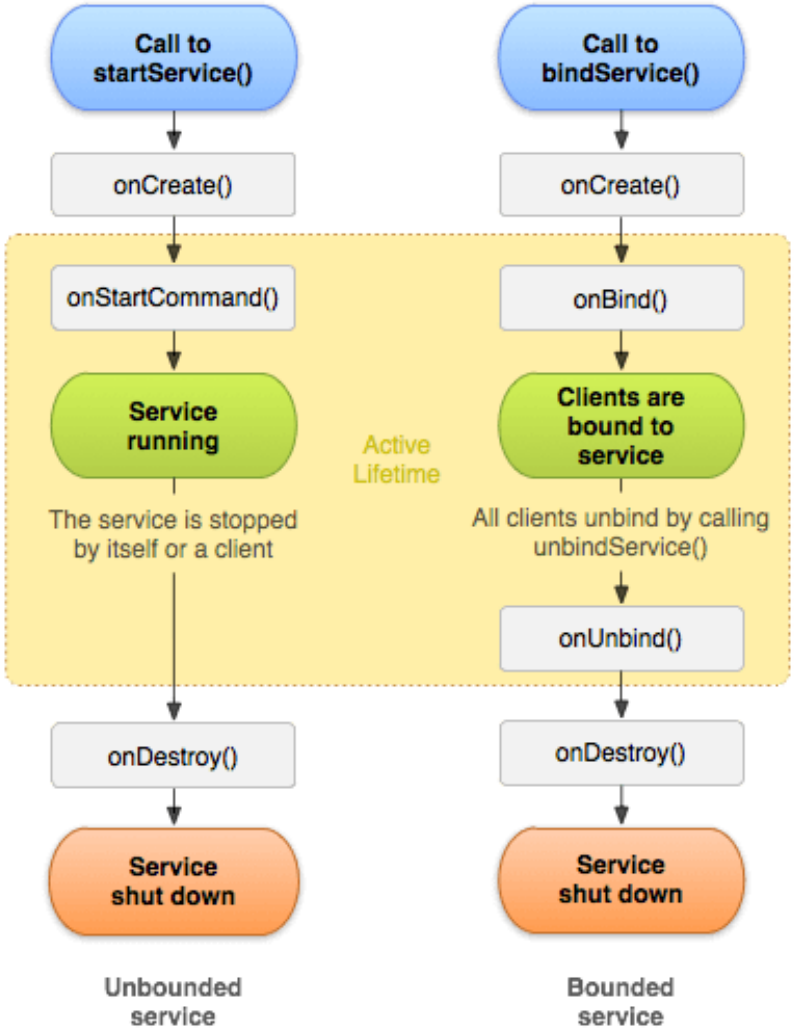
Background

A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

Bound

A service is *bound* when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

The Life Cycle of Android Services



The Life Cycle of Android Services

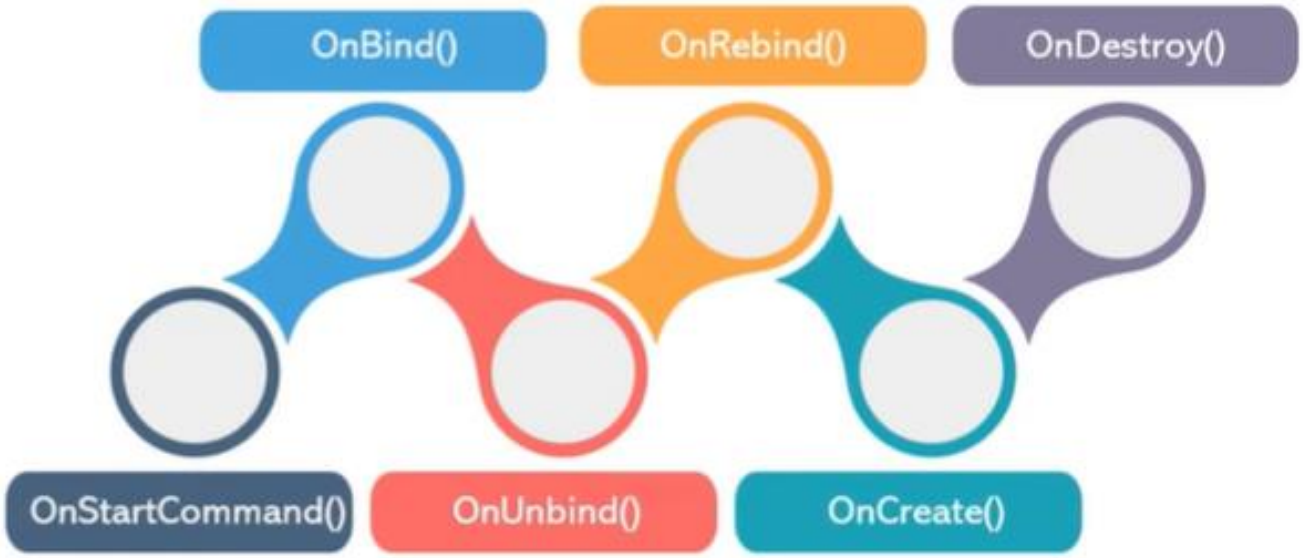
In android, services have 2 possible paths to complete its life cycle namely **Started and Bounded**.

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 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.

Most important callback methods



Most important callback methods

onStartCommand()

The system invokes this method by calling startService() when another component (such as an activity) requests that the service be started. When this method executes, the service is started and can run in the background indefinitely. If you implement this, it is your responsibility to stop the service when its work is complete by calling stopSelf() or stopService(). If you only want to provide binding, you don't need to implement this method.

onBind()

The system invokes this method by calling bindService() when another component wants to bind with the service (such as to perform RPC). In your implementation of this method, you must provide an interface that clients use to communicate with the service by returning an IBinder. You must always implement this method; however, if you don't want to allow binding, you should return null.

onCreate()

The system invokes this method to perform one-time setup procedures when the service is initially created (before it calls either onStartCommand() or onBind()). If the service is already running, this method is not called.

Most important callback methods

onDestroy()

The system invokes 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, or receivers. This is the last call that the service receives.

onUnbind()

The Android 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.

Service behaviour

START_STICKY

- Services are being explicitly managed & long running
- No need to remember state at kill time
- Long running music playing service

START_NOT_STICKY

- Services are being not explicitly managed
- Services are periodically running and self stopping
- Alarm service or server data polling

START_REDELIVER_INTENT

- Services are being explicitly managed
- Restart from previous state at the kill time
- File download

Create an application to demonstrate Android Service (Playing music in background).

MyService.java

```java
package com.example.serviceapp;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.provider.Settings;

public class MyService extends Service {
    private MediaPlayer player;
    public MyService() {
    }
    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        player= MediaPlayer.create(this, Settings.System.DEFAULT_ALARM_ALERT_URI);
        player.setLooping(true);
        player.start();
        return START_STICKY;
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        player.stop();
    }
}
```

activity_main.xml

```
<?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"
    android:layout_margin="20dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="“80dp"
        android:layout_marginTop="“20dp"
        android:text="Services In Android: Playing music"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textColor="@android:color/holo_green_dark"
        android:textSize="36sp"
        android:textStyle="bold" />
    <Button
        android:id="@+id/startButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="“5dp"
        android:layout_marginBottom="“5dp"
        android:backgroundTint="#4CAF50"
        android:text="Start the service"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        android:textColor="#FFFFFF"
        android:textStyle="bold" />
```

```
<Button
        android:id="@+id/stopButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop=“5dp"
        android:layout_marginBottom=“5dp"
        android:backgroundTint="#4CAF50"
        android:text="Stop the service"
        android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        android:textColor="#FFFFFF"
        android:textStyle="bold" />
</LinearLayout>
```

MainActivity.java

```
package com.example.serviceapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.provider.Settings;
import android.view.View;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
findViewById(R.id.startButton).setOnClickListener(new
View.OnClickListener() {
   @Override
   public void onClick(View view) {
Intent i=new Intent(MainActivity.this,MyService.class);
startService(i);

   }
});
```

```
findViewById(R.id.stopButton).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        stopService(new
Intent(MainActivity.this,MyService.class));
    }
});
    }
}
```