Android Services

Android Services

Hüseyin Özkoç · Follow

5 min read · Nov 20, 2021

Hey Folks! In my article today, I will explain Services in Android.

What are Android Services?

Service is the one of the critical application component which can perform long-running operations in the background. It may continue running for time, even user dealing with to another applications. Moreover, main android components can bind to service to interact with it, also you can perform interprocess communication.

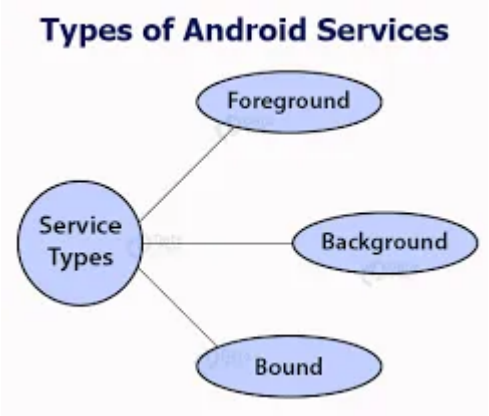
Why we should use services?

As I mentioned before, Developers should use services for long-running operations such as handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Beware! Services run in the **main thread of in hosting process**, **not create its own thread**, so developers should run any blocking operations **on the separate thread(manage yourself)** for avoiding **Application Not Responding** (ANR) errors.

Types of Services

These are the three different types of services:



Android Service Types

1-Foreground:

Type of services that perform operations in the background that is **noticeable for the users.** This kind of services **must display a Notification** and It should continue running even user is not dealing with the app. Likewise, Foreground services **have own lifecycle** that independent from activity or fragment that they were created.
Examples of applications that will use foreground services can be listed as follows:
-**Music Player App**(notification of the current song)
-**Fitness App**(show the distance that user has traveled)

Beware! The WorkManager API offers the flexible and nearly same way as foreground services. In many cases, developers **should prefer using WorkManager** instead of Foreground services.

2-Background:

Type of services that perform operations in the background that is not giving any information to user via notification. Background services **have own lifecycle** that independent from activity or fragment that they were created. **For example**, developers can use background services to **compact its storage.**

Beware! If you targets API level 26 or higher, system **imposes restrictions on running background services.** For example, developers should not access location informations from the background. Alternatively, **schedule tasks using WorkManager.**

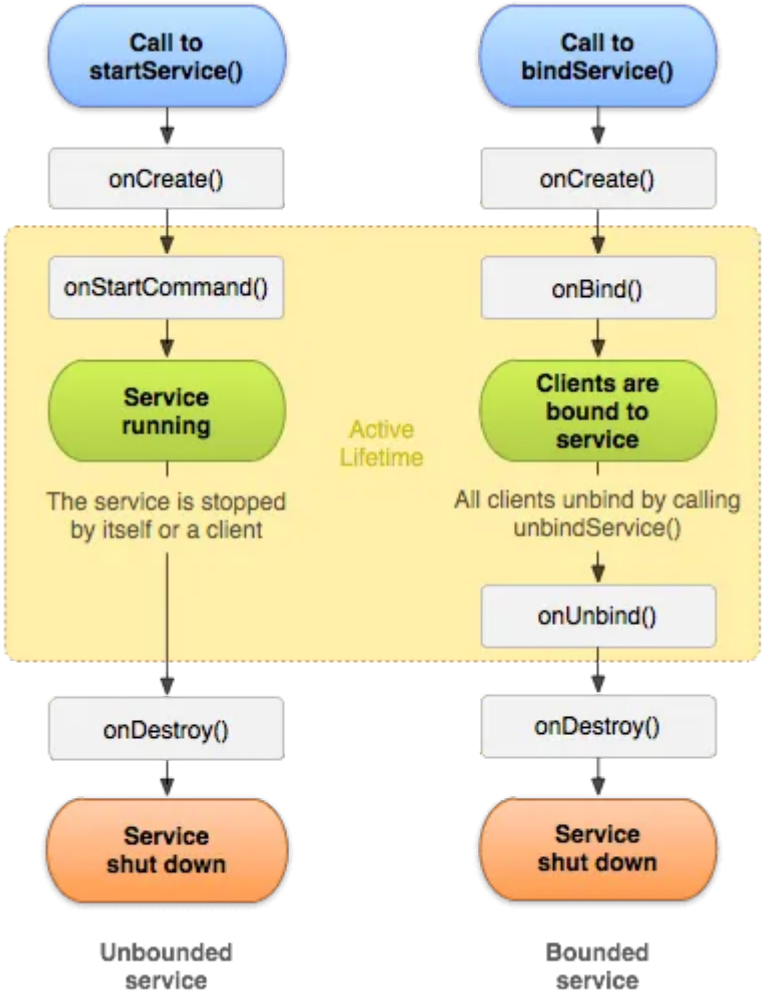
3-Bound:

Type of services that offers a client-server interface that **allows components(Activity, content provider and service can bind to the Bound service) to interact with** the service, send requests, receive results, and even do so across processes with interprocess communication (IPC).

Bound services have **no own lifecycle.**That's why, they use the **lifecycle of the activity or fragment they were bounded.** Moreover, bound services can only run when application component is bounded to it. Likewise, **multiple components** can bind to service at once, but when **all of them unbind,** the service is **destroyed.**

Lifecycle of Service

In the section so far, we talked about how services work, service types and their examples. Now we will review the lifecycles of services to better understand their working structure.



The lifecycle of started service and bound service.

As we can see in the example, we have two kinds of service lifecycle.

1.Started Services

A service becomes started only when an application component calls **startService()**.

Once this service starts, it runs in the background even if the component that created it destroys(**run indefinitely**). This service can be stopped only in one of the two cases:

- By using the **stopService()** method.
- By stopping itself using the **stopSelf()** method.

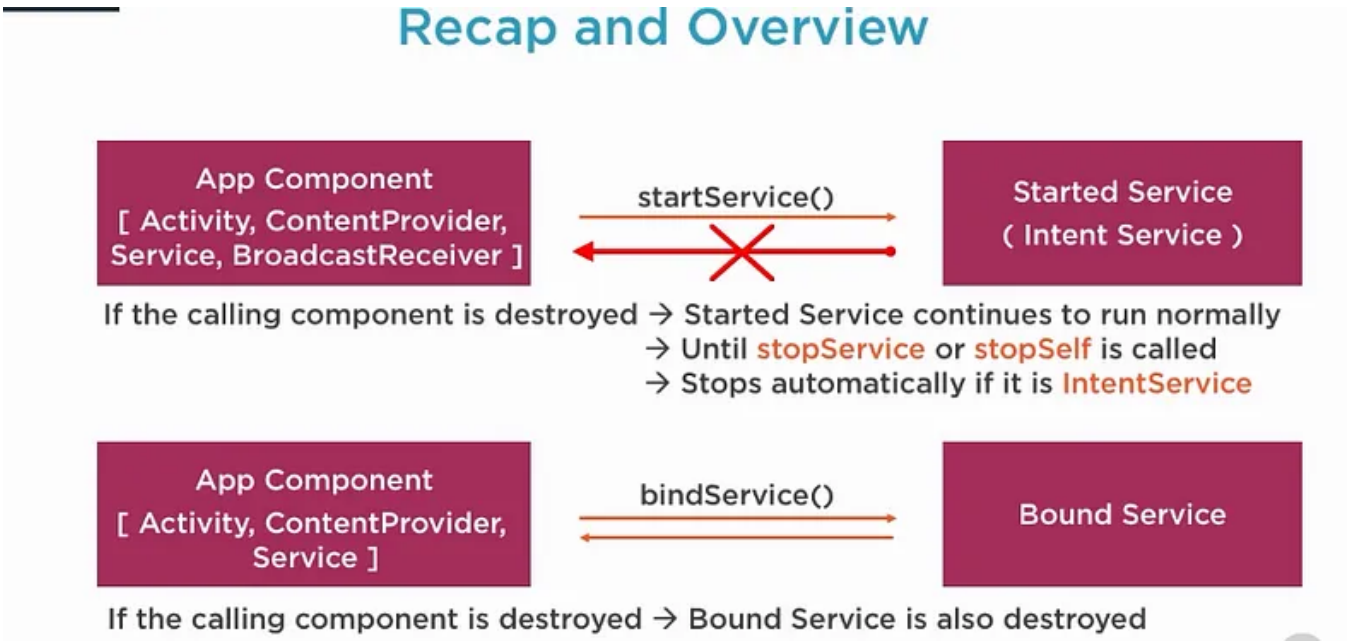
2.Bound Services

A service is bound only if an application component binds to it using **bindService()**.When all clients unbind from bound service by calling **unBindService()** function, service ends with **onUnbind onDestroy** functions.

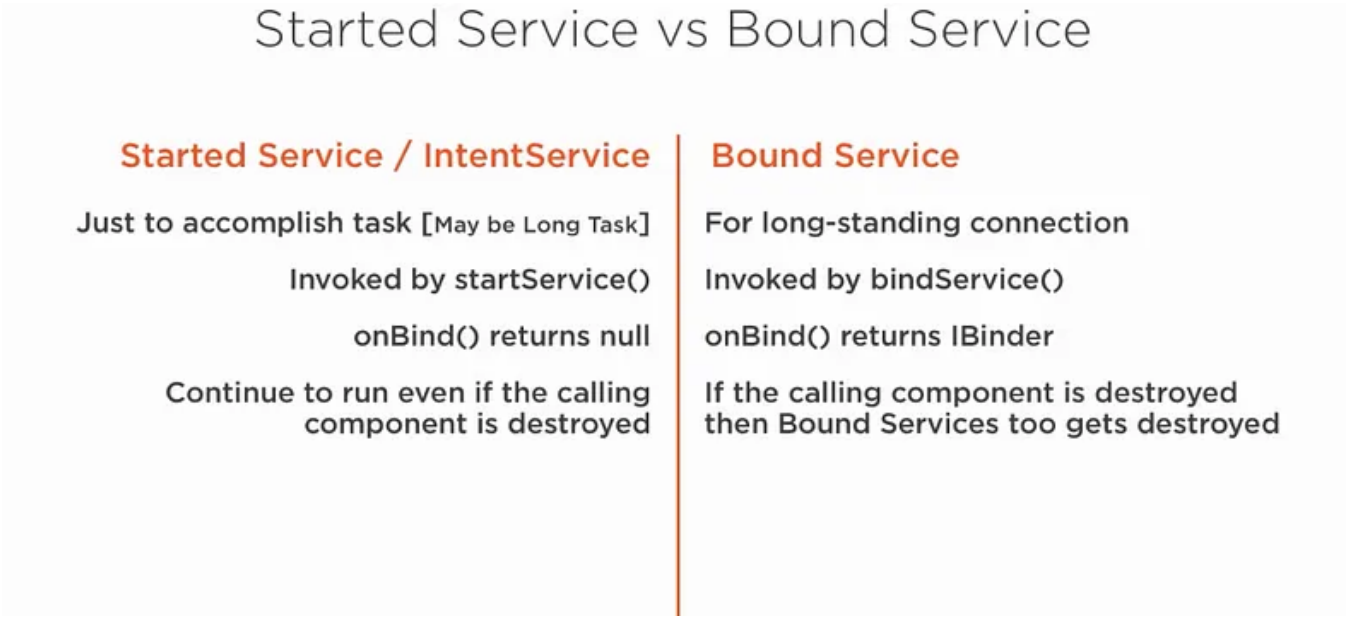
Beware! Although the two types of lifecycles seem separate from each other, our service can work in two ways. **It can be started (to run indefinitely)** and **also allow binding.** It's simply a matter of whether you implement a couple of callback methods: `onStartCommand()` to allow components to start it and `onBind()` to allow binding.

Shortly, our service can b**e both started and bound service at the same time.**

What is the differences between Started Service and Bound Service?



What is the differences between Started Service and Bound Service?



What is the differences between Started Service and Bound Service?

Intent Service

IntentService is an extension of the Service component class that handles asynchronous requests (expressed as Intent) on demand. It handles each Intent one by one with the help of a worker thread(not using main thread) and stops automatically when the work is done. All the requests are controlled using a single worker thread so they may be sometimes time-consuming. However, they do not interfere with the main loop of the application.

> *To use it, **extend IntentService** and implement onHandleIntent(android.content.Intent) . IntentService will receive the Intents, launch a worker thread, and stop the service as appropriate.*

Beware! This class was deprecated in API level 30. Consider using WorkManager or JobIntentService , which uses jobs instead of services when running on Android 8.0 or higher.

Creating a Service

1-First of all, you must declare your service in your application's manifest file.

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

2-Extend your class.

```
public class ExampleService extends Service {
    int startMode;       // indicates how to behave if the service
is killed
    IBinder binder;      // interface for clients that bind
    boolean allowRebind; // indicates whether onRebind should be
used

    @Override
    public void onCreate() {
        // The service is being created
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
{
```

```
        // The service is starting, due to a call to startService()
        return startMode;
    }
    @Override
    public IBinder onBind(Intent intent) {
        // A client is binding to the service with bindService()
        return binder;
    }
    @Override
    public boolean onUnbind(Intent intent) {
        // All clients have unbound with unbindService()
        return allowRebind;
    }
    @Override
    public void onRebind(Intent intent) {
        // A client is binding to the service with bindService(),
        // after onUnbind() has already been called
    }
    @Override
    public void onDestroy() {
        // The service is no longer used and is being destroyed
    }
}
```

I hope it was a useful article for you. I wish you healthy days!