SCIENCES SORBONNE UNIVERSITÉ

Services

Renault@lrde.epita.fr

What is the goal of Services?

Background Tasks

- Similar to services on Linux
- No GUI
- Useful for running a background task
  - **Music for instance**

Lifecycle is simplified

A Service can be controlled by an activity or an widget

- in other words, we (most-of-the-time) needs an UI to interact

Services can be shared among multiple components

More details

Warning!

- By default, the service run in the process that triggered the service
- Neither a thread, nor a separated processus by default!

The system only

- instantiate the service
- run the callbacks
- the developper has to allocate ressources, use dedicated thread, etc.

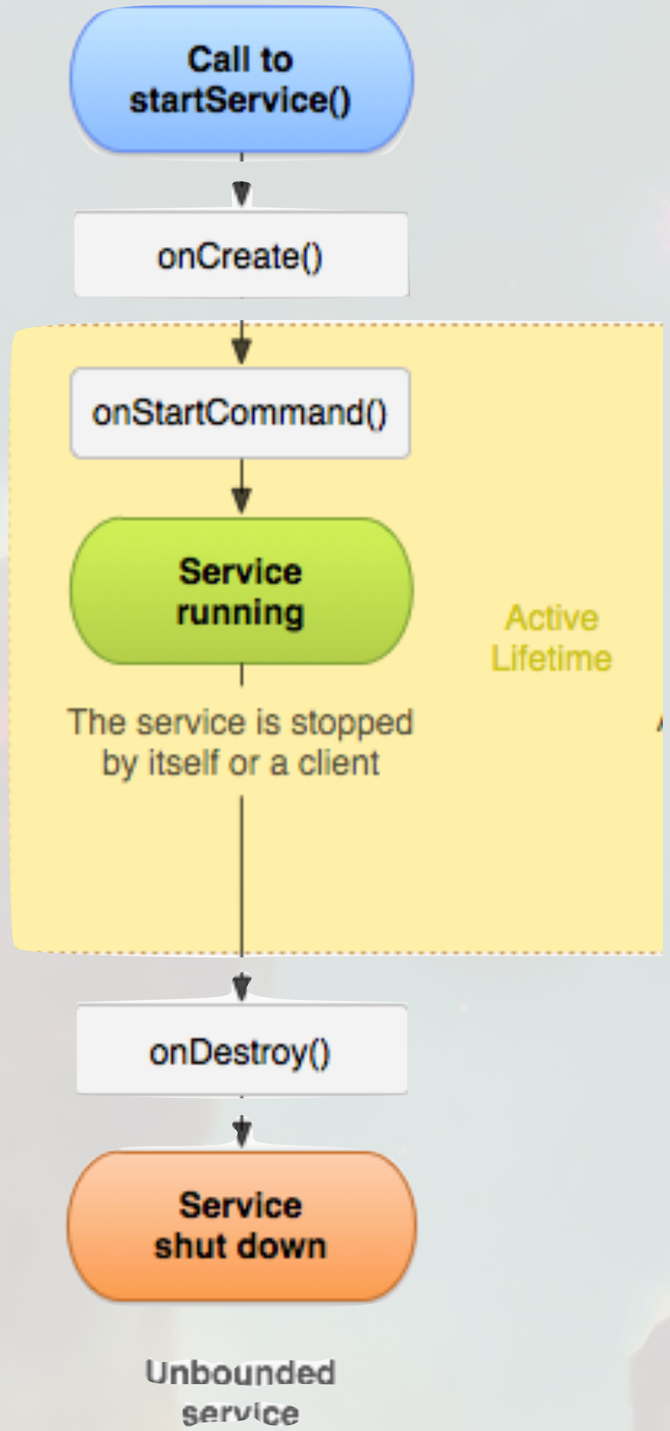
Two kind of services exists

- Bounded:
  - **a single instance**
  - **runs continuously**
- Unbounded:
  - **runs as long as some component use it**

Unbounded Services

Running the service

- onStartService
- Parameters are passed through Intents



One-way communication

Stopping the service

- stopService
- The number of start request is not taking in account for stopping the service

Unbounded Service: Media Player Example (1/2)


```
public class UnboundedService extends Service {
    MediaPlayer mediaPlayer;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate(){
        mediaPlayer = MediaPlayer.create(this, R.raw.mymusic);
        mediaPlayer.setLooping(false);
    }

    @Override
    public void onDestroy(){
        mediaPlayer.stop();
    }
```

Unbounded Service: Media Player Example (2/2)


```
    @Override
    public int onStartCommand(Intent intent, int flags, int
startID){
        mediaPlayer.start();
        // START_STICKY to continue until the
        // service is stopped
        return START_STICKY;
    }
}
```

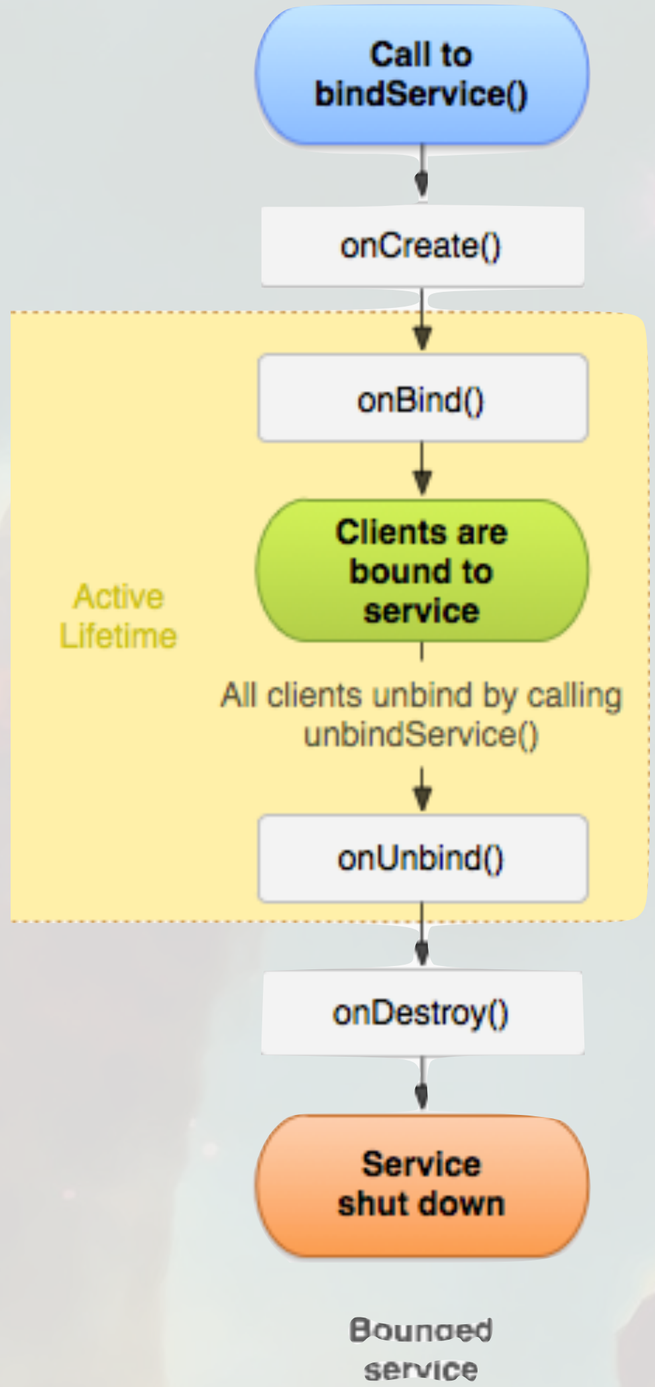
Declaration in the AndroidManifest.xml

```
 <service
        android:name=".UnboundedService"
        android:enabled="true">
</service>
```

Bounded Services

Starting service

- onBind
- returns a binder, close to notion of client/server
- close to IPC
- First client creates the service



The service is manipulated through the binder

Stoping the service

- Clients disconnect
- Last client destroy the service

Bounded Service Example (1/3)

Construct the binder (IBinder)

```
public class IDBinder extends Binder {
    int pseudoRnd = 0;

    public int getNewID(){
        ++pseudoRnd;
        return pseudoRnd;
    }
}
```

Building the service

```
public class BoundedService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return myIDBinder;
    }
    IDBinder myIDBinder = new IDBinder();
}
```

Bounded Service Example (2/3)

Connecting with the bounded service

```
boolean isBinded = false;
    IDBinder idService;
    ServiceConnection mConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name,
                                       IBinder service) {
            idService = (IDBinder) service;
            isBinded = true;
        }
        @Override
        public void onServiceDisconnected(ComponentName name) {
            isBinded = false;
        }
};
```

Bounded Service Example (3/3)

Fix the AndroidManifest.xml

```
<service android:name=".BoundedService" ></service>
```

Connect to the service

```
getApplicationContext()
    .bindService(new Intent(getApplicationContext(),
                                BoundedService.class),
                 mConnection, BIND_AUTO_CREATE);
```

Disconnect to the service

```
getApplicationContext().unbindService(mConnection);
```

Call the service

```
idService.getNewID();
```

Summary

Two kind of Services

- Bounded or not depending on your needs
- Can be started with different policies in case of early terminaison (START_STICKY)

Services work well with widgets to build background tasks

Services have their own lifecycle

Must handle threads

- in the example, music player handle that