

# Services & Alarms

Dr Tadhg O'Sullivan

# Outline

- Services
- Alarm Manger
- Which one to choose and why ?
- Service example
- IntentService
- Foreground Services
- Alarms Exact and Inexact
- Alarm example

# Service

- An application component that can perform long-running operations in the background and does not provide a user interface
- Started manually via API, or via IPC
- Runs within the main thread of an application by default, so creating a new thread (with `HandlerThread`) or using `AsyncTask()` is often needed!

# Alarm Manager

- Allows you to schedule your application to run at set point in the future.
- When activated it will call an intent that has been previously registered with it.
- Works even if your application is not running
- Alarms can exact or inexact

# Do You Really Need a Service/Alarm ?

- Services run continuously, draining battery, so use sparingly!
- If you only need to perform something in a background while user interacts with your app, just use threads!
- Prefer using (Inexact) Alarms and Intent Receivers instead, if possible!

# Service Types

- Started
  - A service is "started" when an application component (such as an activity) starts it by calling **startService()**
- Bound
  - A service is "bound" when an application component binds to it by calling **bindService()**

# Started Service

- Started by **startService()**
- Runs in background indefinitely until it stops itself with **selfStop()** or stopped by **stopService()**
- Separate from activity that started it
- Does not return any result to the caller
- Identified by implementing **onStartCommand()**

# Bound Service

- Bound when called with `bindService()`
- Offers client-server interface allowing interaction, get requests, send results
- Identified by implementing `onBind()`
- Runs as long as another application component is bound to it (can be multiple).  
When all unbind, the service is destroyed

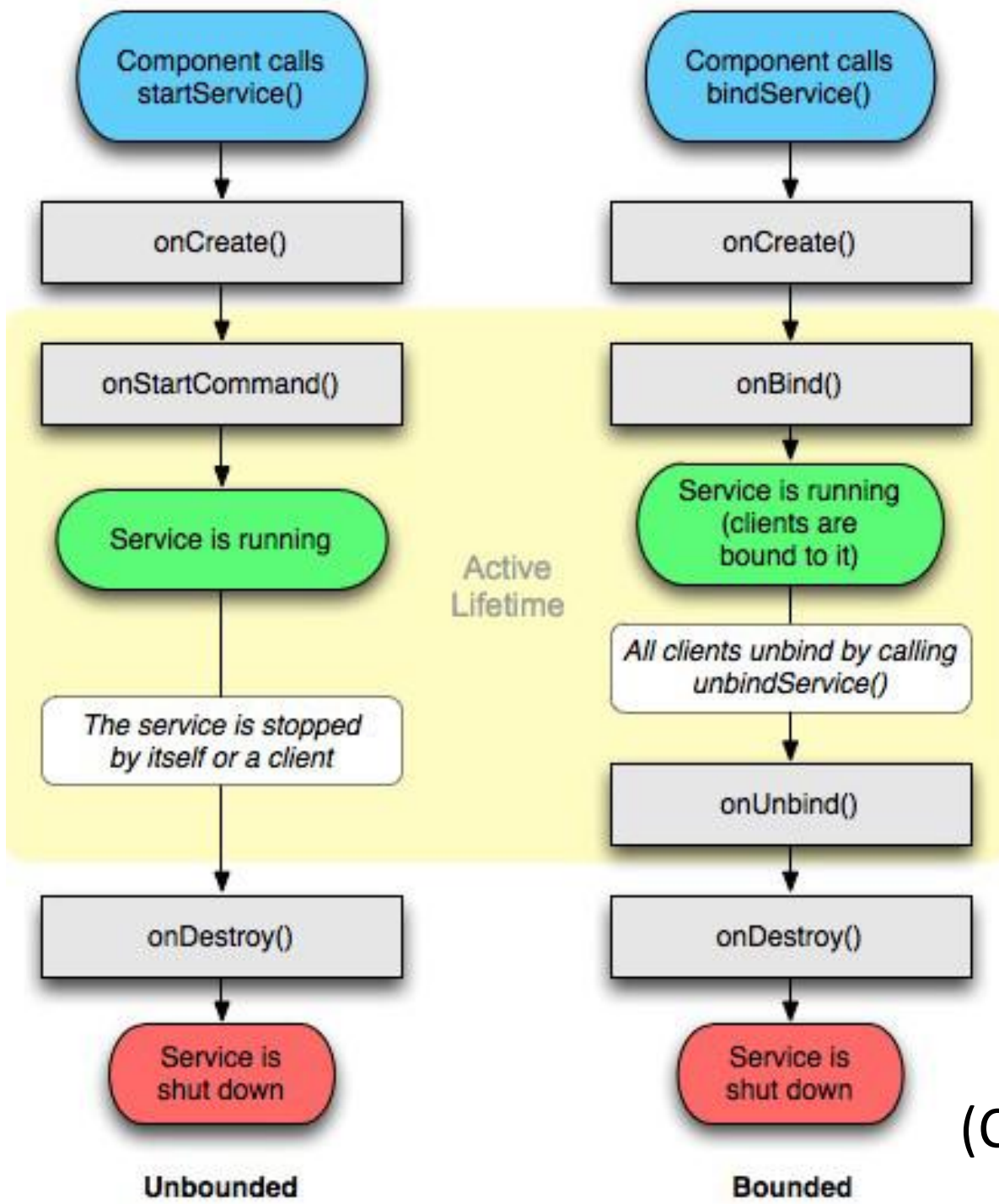


# Services Cont'd

- Services can be both started and bound at the same time:
  - Implemented both **onStartCommand()** and **onBind()**
  - Requires both **serviceStop()** or **selfStop()** AND all app component to unbind to be destroyed
- By default service runs within the main thread of the application! So create threads or use AsyncTask

# Service Methods

- **onCreate()**
- **onBind()**
- **onStartCommand()**
- **onDestroy()**



# Service Definition

```
<service android:name=".OurService">  
  <intentfilter>  
    <action android:name=  
      "ie.ucd.START_OUR_SERVICE" />  
    <category android:name=  
      "android.intent.category.DEFAULT" />  
  </intentfilter>  
</service>
```

# Service Example

```
public void onStartCommand(Intent i, int
flags, int startId)
{
    MediaPlayer p;
    super.onStartCommand(i, flags,
startId);
    player = MediaPlayer.create(this,
R.raw.test);
    player.start();
}
```

# Service Example Cont'd

```
public void onDestroy()  
{  
    super.onDestroy();  
    player.stop();  
}
```

# Service Example Cont'd

- `startService(new Intent(this, "ie.ucd.START_OUR_SERVICE"));`
- `stopService(new Intent(this, "ie.ucd.START_OUR_SERVICE"));`

# IntentService

- Quick and easy alternative to a full service
- Does create a separate thread to execute all intents it receives
- Does not support multiple simultaneous requests (queues them instead)
- Implements `onHandleIntent()` and does the work there
- Dies after handling all intents it receives



# Foreground Services

- You can mark a service as 'Foreground', telling the system your user is aware of this service and actively interacts with it
- Use 'startForeground()' and 'stopForeground()' to manage service state
- Foreground services aren't killed by the system, but require active notification in status bar

# Alarm Manager

- Can be set to **setExact()** to set alarm at a specific time.
- Or to be more energy efficient it can be **set()** but this will be inexact as it's waiting for a time when the phone will check, worst case scenario could be up to the full length of the alarm , e.g. an hourly alarm could be off by nearly an hour.

# Alarm Manger Cont'd

- If you need to setup or check for an alarm make sure its restarted at boot in case the phone has been switched off.
- Using an intent-filter and a broadcast receiver they can be setup after boot  
(*android.intent.action.BOOT\_COMPLETED*)

# Alarm Manger Example Manifest

```
<receiver
  android:name=".boot.RunOnBootToSetupAlarms"
  >
<intent-filter>
  <action
    android:name="android.intent.action.BOOT_COMPLETED" />
</intent-filter> </receiver>
```

# Alarm example

```
Intent intent2 = new Intent(context, SetupNextAlarm.class);
PendingIntent sender = PendingIntent.getBroadcast(context, 0, intent2, 0);

    long everyHour= 3600000 ; // (an hour in milliseconds)
    long firstTime= 3600000 ; // (an hour milliseconds)

        AlarmManager am =(AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);

am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                firstTime, everyHour, sender);
```

# Questions

- Please ask in the Student Forum