Running in the background: Services

Joscha Behrmann <behrmann@hm.edu>
Salman Alhadziev <salman.alhadziev0@hm.edu>

Overview

- What is a service?
- How do I declare and start a service? (ex. 1)
- What is the lifecycle of a service?
- What would I want to use a service?
- A practical example of a bound service. (ex. 2)
- Service types
- Foreground service demo. (ex. 3)
- Summary & Questions

What is a service?

The general case

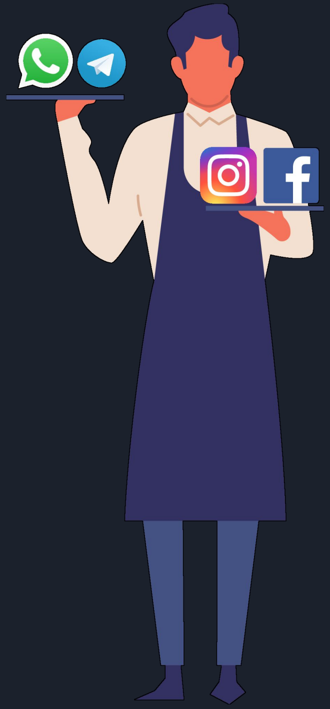
"A background process is a computer process that runs behind the scenes (i.e., in the background) and without user intervention. Typical tasks for these processes include logging, system monitoring, scheduling, and user notification. The background process usually is a child process created by a control process for processing a computing task." [1]

The Android case

“A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use.“ [2]

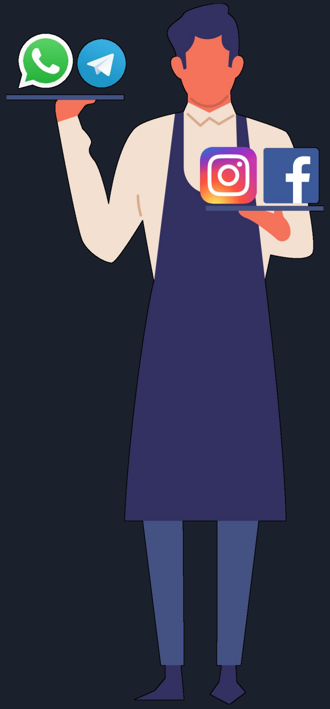
Services in Android (1)

- Omnipresent
- (mostly) Invisible to the user
- Used to perform long-running operations in the **background**
  - Play music
  - Fetch data over the network
  - Perform file I/O
  - Long-running calculation (e.g. image transformations)
- No user interface
  - Don’t (directly) interact with the UI



Services in Android (2)

- Declared as *<service>* in the AndroidManifest.xml
- Started through *Context.startService()* or *Context.bindService()*
- Run in the main-thread
  - Not a separate process
  - Not a separate thread
- Tell the system that the application wants to…
  - be doing something in the background (*startService*)
  - expose some functionality to other apps (*bindService*)



How do I declare and start a Service?

Minimum Working Example (MWE) (1)

```xml
<!-- AndroidManifest.xml -->

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

    <application>
        <... />

        <service android:name=".MyExampleService" />

        <... />
    </application>

</manifest>
```

Minimum Working Example (MWE) (2)

```kotlin
/* MyExampleService.kt */

class MyExampleService : Service() {

    override fun onCreate() {
        super.onCreate()
        Log.i(MyExampleService::class.simpleName, "onCreate called")
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        Log.i(MyExampleService::class.simpleName, "onStartCommand called")
        return super.onStartCommand(intent, flags, startId)
    }

    override fun onBind(intent: Intent?): IBinder? {
        return null
    }
}
```

Minimum Working Example (MWE) (3)

```kotlin
/* MainActivity.kt */

override fun onCreate(savedInstanceState: Bundle?) {

    /* ... */

    val intent = Intent(this, MyExampleService::class.java)
    startService(intent)

    /* ... */
}
```

Minimum Working Example (MWE) (4)

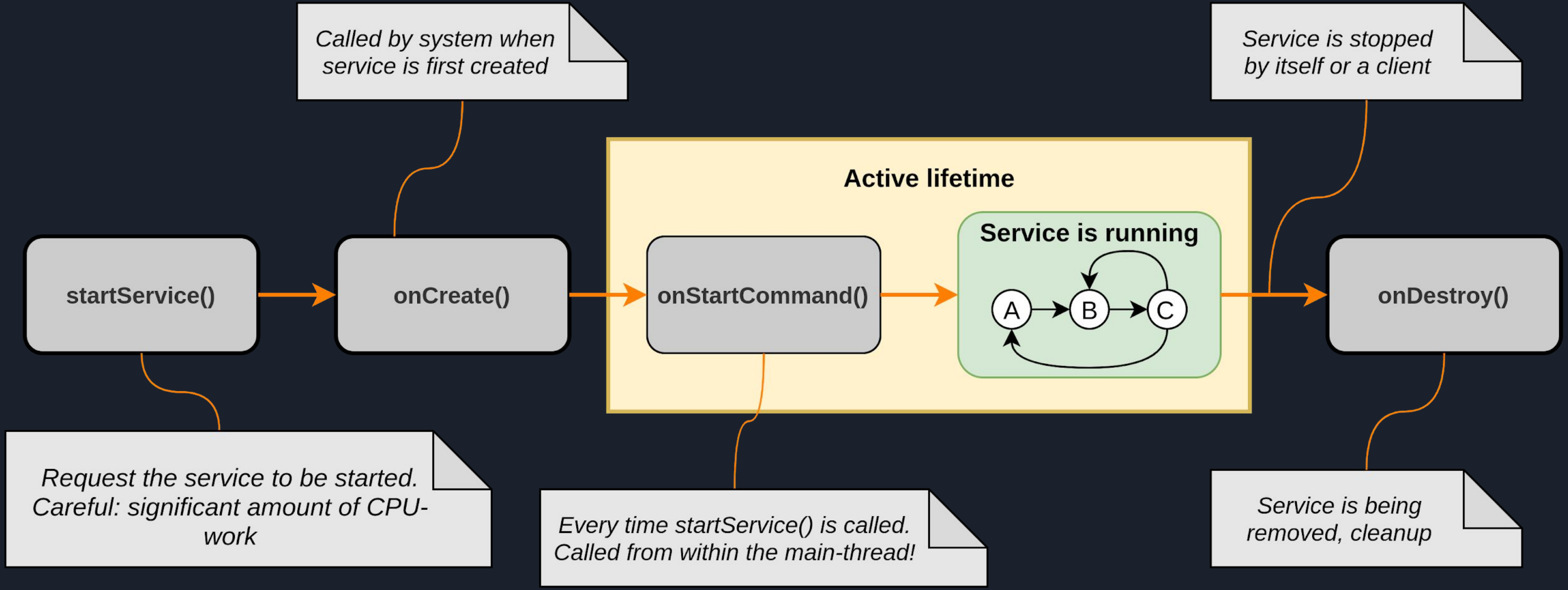
```
2020-05-21 16:04:36.627 I/MyExampleService: onCreate called
2020-05-21 16:04:36.627 I/MyExampleService: onStartCommand called
```

First observations

- We created a service that just writes to the log
- Creating the service was simple
  - Declare in Manifest
  - Create class that extends *Service*
  - Start the service from an activity
- Services have a **lifecycle**

What is the lifecycle of a service?

Lifecycle of a (unbounded) service


Called by system when service is first created
Service is stopped by itself or a client
Active lifetime
Service is running
startService()
onCreate()
onStartCommand()
A
B
C
onDestroy()
Request the service to be started. Careful: significant amount of CPU-work
Every time startService() is called. Called from within the main-thread!
Service is being removed, cleanup

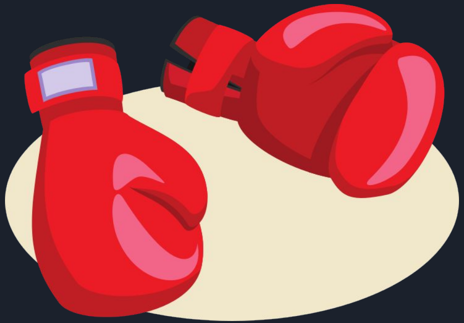
Lifecycle of a service

- Several more callbacks [2]
- Callbacks can be used to implement the services features
- For bound services
    - onBind()
    - onUnbind()

Why would I want to use a Service?

Services vs. Threads

- Services can run in the background
  - Still run when the user is not interacting with the app
  - No coupling UI : Service
- Threads perform work outside of the main-thread
  - Only while user is interacting with the app
  - Tight coupling UI : Thread
  - Make use of the hardware capabilities

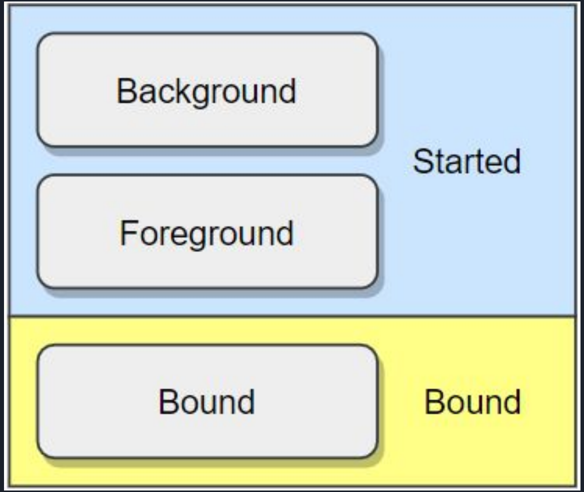


Services with Threads

- User opens the app, a process is created
- Starts with a single “main thread” (UI)
    - Responsible for everything that happens on-screen
    - Takes and executes work from a queue
    - User-events, lifecycle-callbacks
- 60 fps target (16ms/frame)
    - Smooth user-experience

A practical example of a bound service.

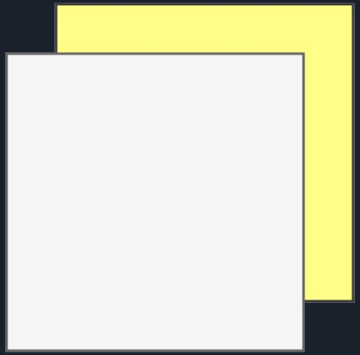
Service Types



- Three Types
  - Background
  - Foreground
  - Bound
- Or two Types
  - Started
  - Bound
- Service can be both

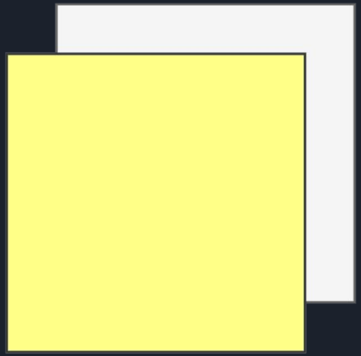
Background Service

- Performs operations that are noticeable to the user
  - Compact storage of the app
- Can be started by other app component using startService()
- Runs indefinitely
  - Until stopSelf() or stopService() is called
- Likely to be killed by system when memory is low
- Restart configured by return value in onStartCommand()



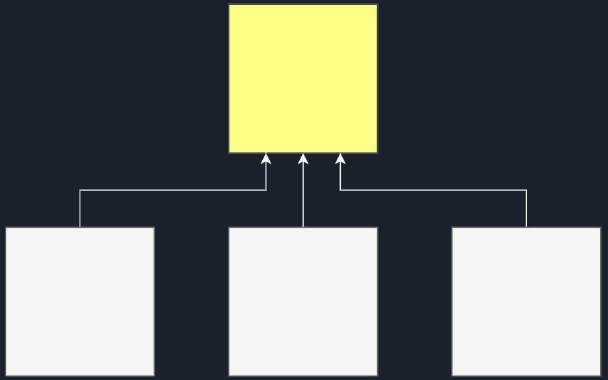
Foreground Service

- Performs operations that are noticeable to the user
  - Play music
- Can be started by other app component using startForegroundService()
- Needs to be done immediately
- Needs to be executed to completion
- Rarely killed by system
- Must display a Notification



Bound Service

- Offers a client-server interface
  - Interact with service
  - IPC
- Components can bind to the service using bindService() and unbind using unbindService()
- Multiple components can bind to one service at once
- Runs as long as another component is bound to it
- Must implement onBind() and return a IBinder



Foreground Service Demo.

Summary

- Services for long-running operations in the background
- Threads vs. Services
- No user interface
- Run in the main thread by default
- Three types of services
  - Background
  - Foreground
  - Bound

Questions?

References

[1] Proceedings of ICACNI 2018, Volume 1. “Smart Computing Paradigms: New Progresses and Challenges”. Springer Fachmedien GmbH

[2] “Android Developers Documentation: Service”, [Online] Available: https://developer.android.com/reference/android/app/Service [Accessed 21 May 2020]