Services

https://developer.android.com/guide/components/services.html


Repo: https://github.com/karlmorris/AndroidService/
Overview

- Purpose of services
- Types of service
- Service lifecycle
- Implementation examples
Services

- Services provide long running processes in the background
- Does not have a user interface (no layout is attached)
- Components **bind** to services to interact with them
Services cont.

- Started by a component using startService()
- Usually performs a single operation
- Service should stop itself when complete
Examples of using Services

- Playing music
- Uploading or downloading large files
- Interacting with external content providers
Service states

- **Started**
  - When a component calls `startService()`
  - Remains in **started** state even when calling component is destroyed
  - Stopped using `stopSelf()` or `stopService()`
- **Bound**
  - When a component calls `bindService()`
  - Offers client-server interface to allow interaction from components
  - Multiple components can be bound simultaneously
  - Automatically destroyed when last component unbinds
Service Lifecycle

- **Call to `startService()`**
  - `onCreate()`
  - `onStartCommand()`
  - **Service running**
    - The service is stopped by itself or a client
    - `onDestroy()`

- **Call to `bindService()`**
  - `onCreate()`
  - `onBind()`
  - **Clients are bound to service**
    - All clients unbind by calling `unbindService()`
    - `onUnbind()`

- **Unbounded service**
- **Bounded service**

---

**Active Lifetime**
Service superclass

- onStartCommand()
  - Called when service is started using startService()
- onBind()
  - Called when service is started using bindService()
- onCreate()
  - Called when service is first created for one time setup
  - Not called if service is already running
- onDestroy()
  - Called before service is destroyed
  - Use to clean up any unwanted listeners, threads, etc.
Declaring a service in the Manifest

<manifest ... >

...  

<application ... >

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

    ...

</application>

</manifest>
Starting a service

Intent intent = new Intent(context, ExampleService.class);
startService(intent);
Stopping a service

• It's important to stop a service when its job is complete

• `stopService()` will cause Android to stop the service as soon as possible
  – Not always ideal/safe

• `stopSelf(int)` is safe for multiple requests
  – `startId` received in `onStartCommand()`
Example: IntentService

- Very simple service implementation
- Subclass of Service
- Handles on-demand asynchronous tasks
- Received intents are processed on a single worker thread (one at a time)

https://github.com/karlmorris/AndroidService
Bound Services

- A service that acts as a server
- Can service multiple requests
- Handles its own lifecycle (typically)
  - Starts when the first client binds
  - Stops itself when the last client disconnects
Bound Service

• To allow binding, must implement `onBind()`
• Returns a `IBinder` object that describes the interface
• Clients connect by calling `bindService()`
Methods of creating Bound Services

- Extend the Binder class
  - Private services
  - Runs in same process
- Use a Messenger
  - Works across different processes
  - Single request thread – simplest IPC
- AIDL (Android Interface Definition Language)
  - For multithreaded IPC
Example: Bound Service

- Service starts on first call to bindService()
- Remains running as long as a client is connected
  - Shuts down on last client disconnecting
- Utilizes IBinder to expose all public methods as interface

https://github.com/karlmorris/AndroidService/tree/Bound