Android Service

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

Topics

- What is a Service?
- Service lifecycles
- Permissions
- Local services
- Remote services

What is a Service?

When a Service Used?

- Applications will need to run processes for a long time without any intervention from the user, or very rare interventions.
- These background processes need to keep running even when the phone is being used for other activities / tasks.

Two Types of Services

- Local service
  - Not accessible from other applications
  - Runs in the same process of the application that starts it
- Remote service
  - Accessible from other applications as well
  - Exposes to other applications through AIDL (Android Services Definition Language)

Service and Notification

- Service is a long lived component and does not implement any user interface of its own.
  - Typically a user interacts with a service through an activity that has a UI.
- The service “notifies” the user in case of any need for user intervention.
  - Through the notification (such as status bar notification), the service can give a user interface for binding to the service or viewing the status of the service or any other similar interaction with the service.

What Facilities Service Provide?

- A facility for the application to tell the system about something it wants to be doing in the background (even when the user is not directly interacting with the application).
  - Through *Context.startService()*, which ask the system to schedule work for the service, to be run until the service or someone else explicitly stop it.
- A facility for an application to expose some of its functionality to other parts of the application or other applications
  - Users of the service calls *Context.bindService()*, which allows a long-standing connection to be made to the service in order to interact with it.

Binding to a Service

- Once a service has begun and is running in the background, activities can “bind” to the service.
  - In fact, if we want to bind to a service that has not started, calling to bind could initiate the service.

Service Lifecycle

Starting a Service

- If someone calls *Context.startService()* then the system will
  - Retrieve the service (creating it and calling its *onCreate()* method if needed) and then
  - Call *onStartCommand(Intent, int, int)* method of the service with the arguments supplied by the client.
- The service will at this point continue running until *Context.stopService()* or *stopSelf()* is called.

Modes of Operations

- For started services, there are two additional major modes of operation they can decide to run in, depending on the value they return from *onStartCommand()*:
  - START_STICKY is used for services that are explicitly started and stopped as needed,
  - START_NOT_STICKY or START_REDELIVER_INTENT are used for services that should only remain running while processing any commands sent to them.

Getting a Connection to a Service

- Clients can also use *Context.bindService()* to obtain a persistent connection to a service.
  - This likewise creates the service if it is not already running (calling onCreate() while doing so), but does not call onStartCommand().
- The client will receive the *IBinder* object that the service returns from its *onBind(Intent)* method, allowing the client to then make calls back to the service.

Running Services User Interface

“Running Services” User Interface

- When invoked, displays a list of all running services that may be of interest to the user, organized by the processes they run in

Why “Running Services” User Interface

- Provided by Android system to control the services running in the system because the use of services has increasing impact on the user experience
  - Too many services consume resources of the device

Permissions

Permissions

- Global access to a service can be enforced when it is declared in its manifest's *<service>* tag.
- By doing so, other applications will need to declare a corresponding *<uses-permission>* element in their own manifest to be able to start, stop, or bind to the service.
- In addition, a service can protect individual IPC calls into it with permissions, by calling the *checkCallingPermission(String)* method before executing the implementation of that call.

Local Service Sample

Local Service

- One of the most common uses of a Service is as a secondary component running alongside other parts of an application, in the same process as the rest of the components.
  - All components of an .apk run in the same process unless explicitly stated otherwise, so this is a typical situation.
- When used in this way, by assuming the components are in the same process, you can greatly simplify the interaction between them
  - Clients of the service can simply cast the IBinder they receive from it to a concrete class published by the service.

Remote Messenger Service

Remote Messenger Service

- If you need to be able to write a Service that can perform complicated communication with clients in remote processes (beyond simply the use of Context.startService to send commands to it), then you can use the Messenger class instead of writing full AIDL files.

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"