Performing Background Work with Services

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT
@hedgehogjim blog.jwhh.com

What to Expect from This Module

Activity-based Background Work Limitations

Background Work with Services

Implementing a Service

Implementing a Service with IntentService

Starting a Service

Starting a Service with PendingIntent

Background Work and Activities

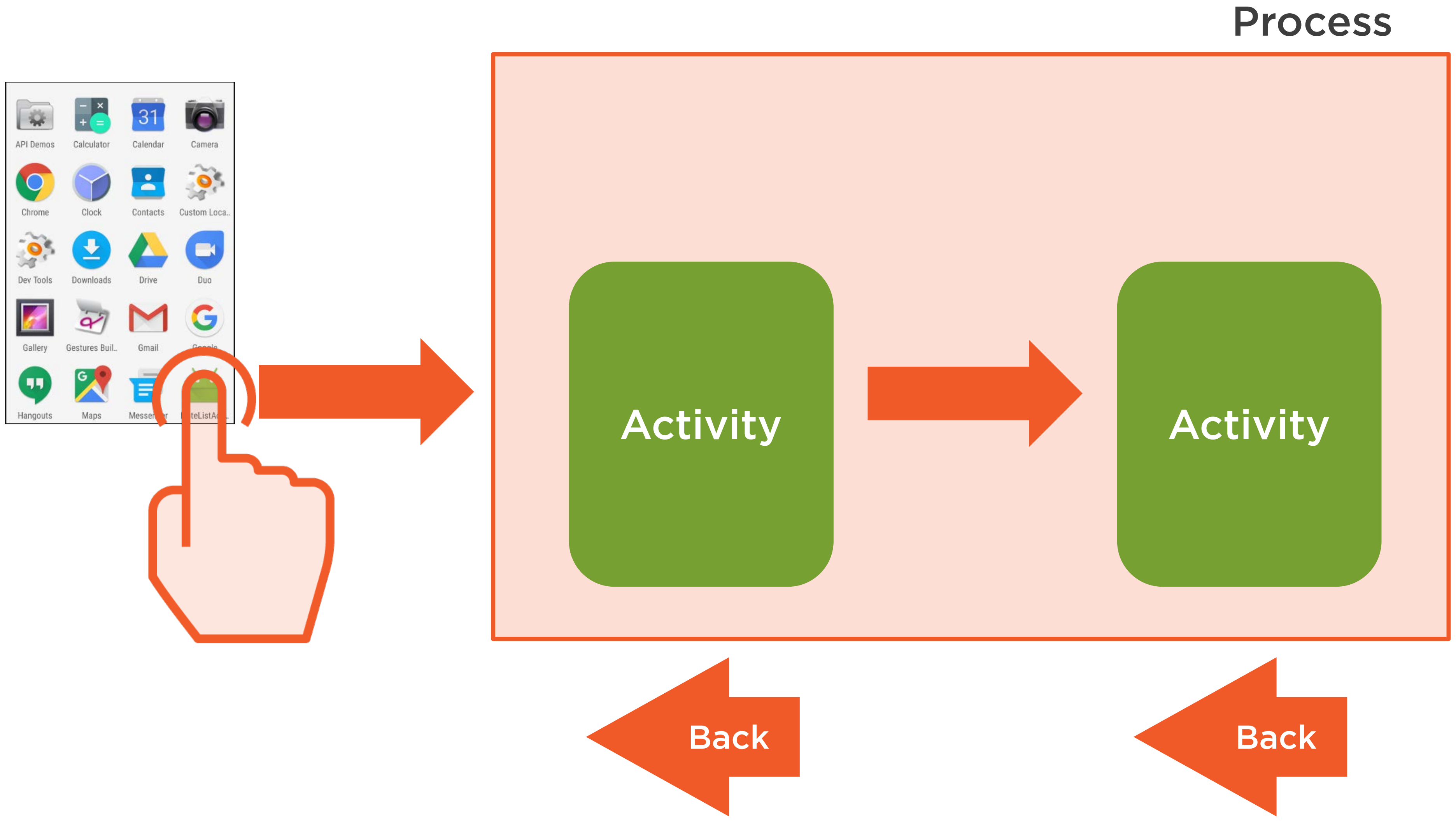
Activities can initiate background work

- CursorLoader, AsyncTask, etc.

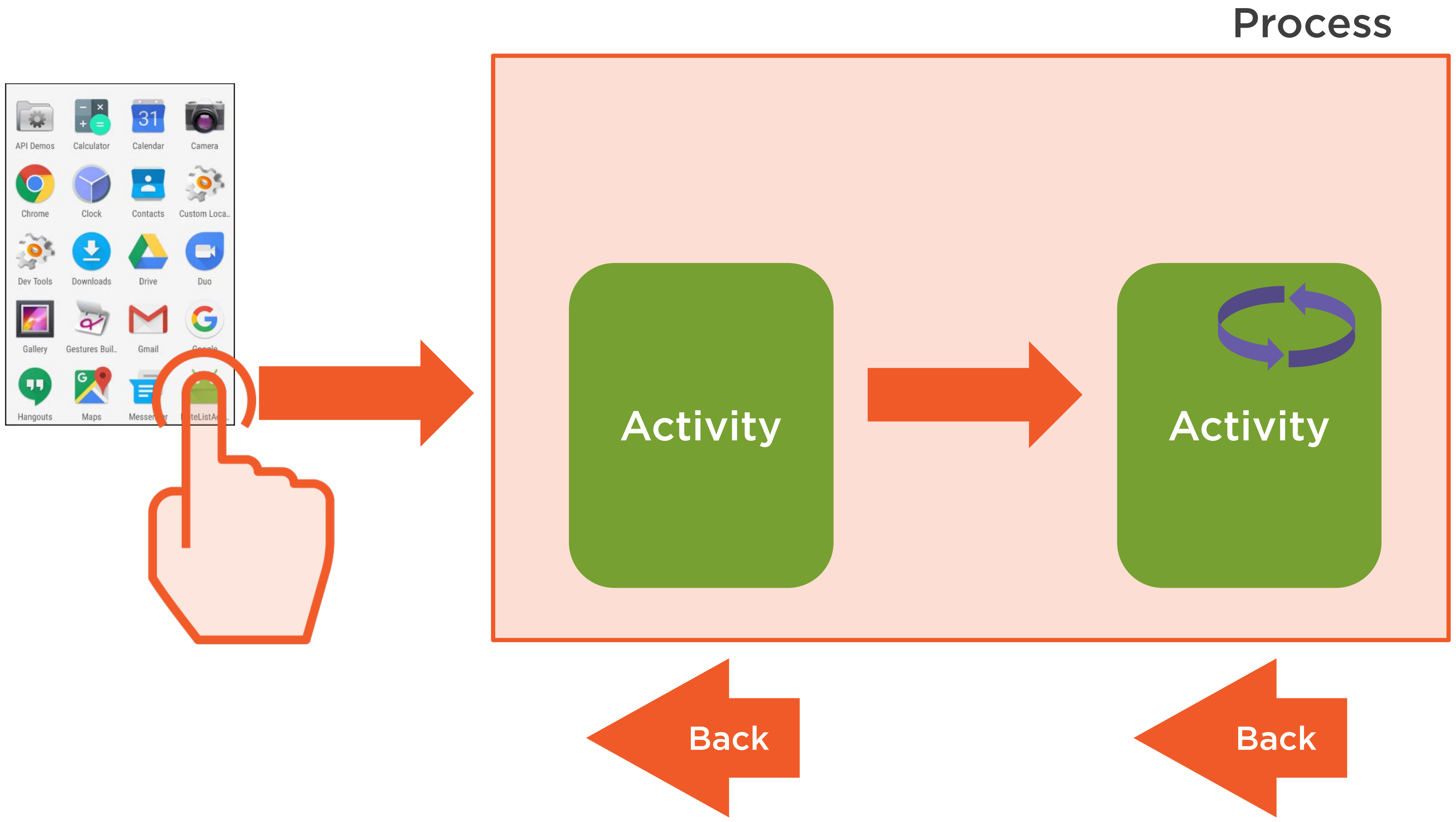
Activities have a lifetime

- Lifetime tied to user interaction
- Can impact background work lifetime
- Background thread may get cleaned up before work is complete

Background Work and Activities



Background Work and Activities

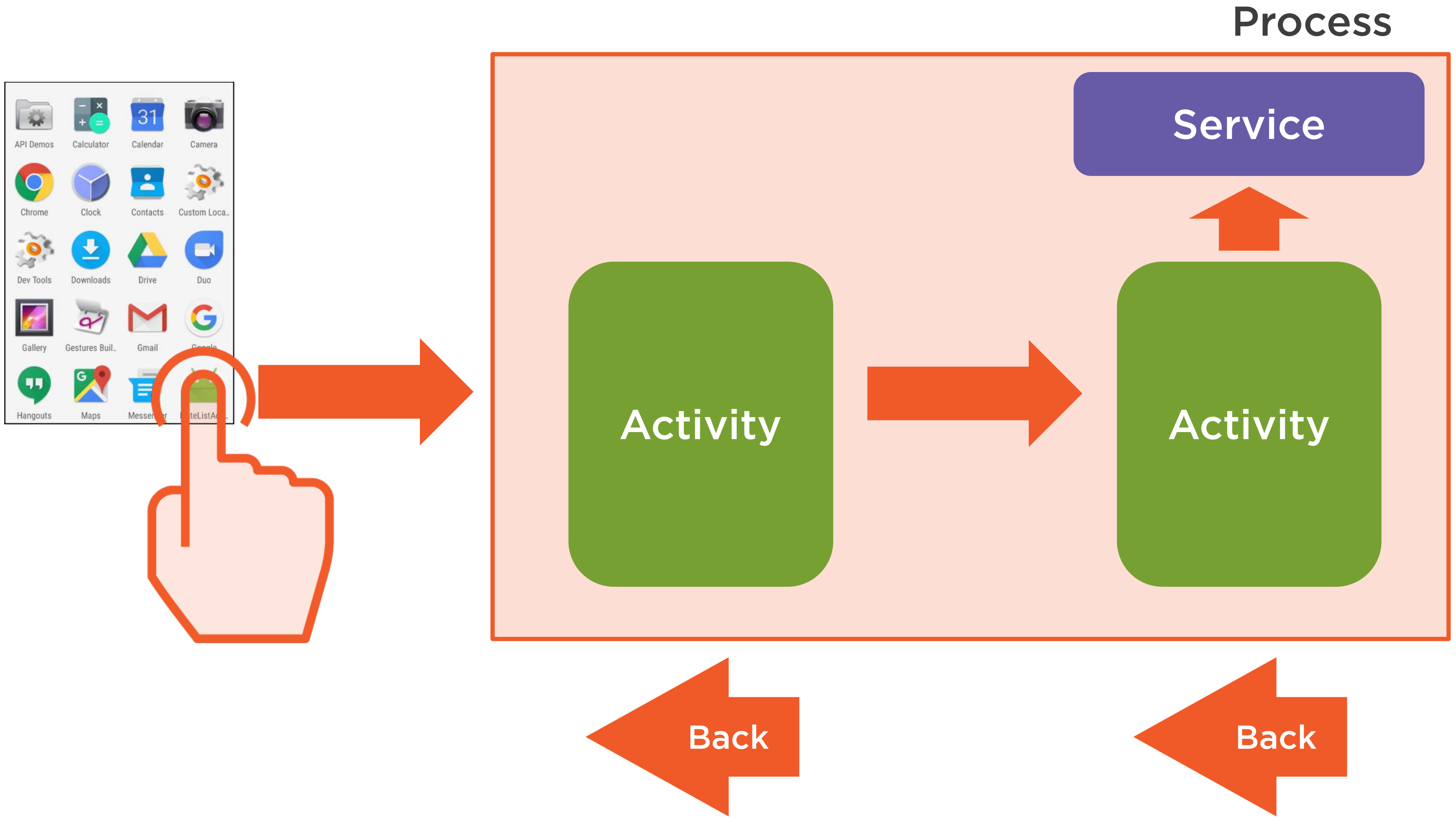


Background Work and Services

Services perform non-UI work

- Makes Android aware that we're doing meaningful work

Background Work and Services



Service

Android is a component-oriented platform

- A number of different types
- Activities are the most familiar

Service is an Android component

- Has a lifecycle
- Does not-present a UI

Service

Perform long-running background work

- Use for work longer than a few seconds
- Continues running even if user switches to another app

Submit work to a Service with an intent

- Create Intent similar to activity intent
- Associate any needed extras
- Pass intent to Context.startService

Implementing a Service

Services extend the Service class

- Provides lifecycle methods
- Provides method to receive work
- Developer left to handle a lot of details

Implementing a Service

Threading behavior

- Work received on main thread
- Need to dispatch work to different thread

Handling of multiple work submissions

- System will start service when needed
- Limited to one running instance at a time
- Additional work submissions sent to that running instance

Implementing a Service

Service lifetime

- Determine when to shutdown
- Determine how to behave when shutdown by the Android system

Implementing a Service with IntentService

IntentService class

- Simplifies service implementation
- Works well for most common scenarios

Implementing a Service with IntentService

Threading issues

- Creates a background LooperThread
- Work performed on LooperThread

Dealing with multiple work submissions

- Work is queued to MessageQueue
- Submission performed one at a time
- Submission performed in order received

Service lifetime

- Shuts down when work complete
- And no more work in queue

Implementing a Service with IntentService

Extend IntentService class

- Call base class constructor
- Pass constructor service name
- Override appropriate methods

Implementing a Service with IntentService

Override onHandleIntent

- Receives intent passed to startService
- Perform service work in this method
- Runs on background LooperThread

Can override other Service methods

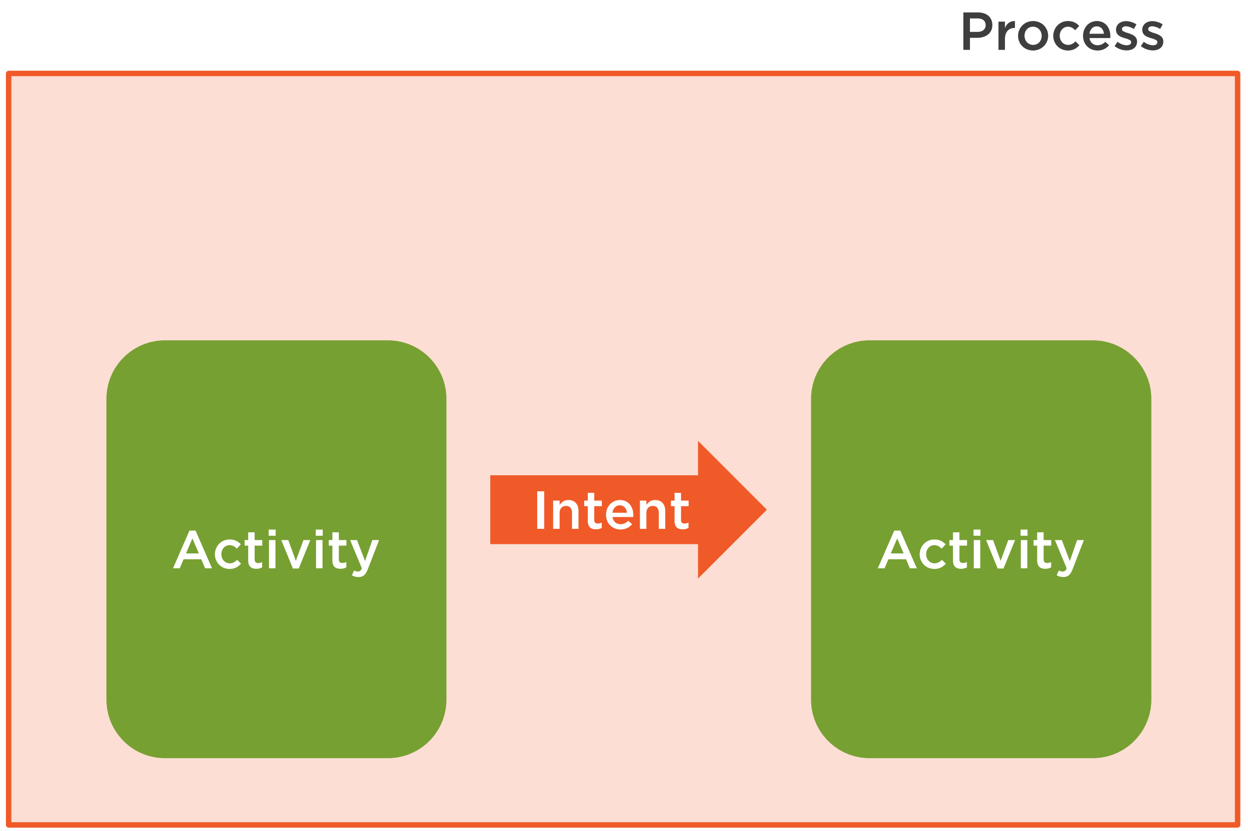
- Be sure to call base implementation
- Helpful when specific work needed at service creation, destruction, etc.

Starting a Service

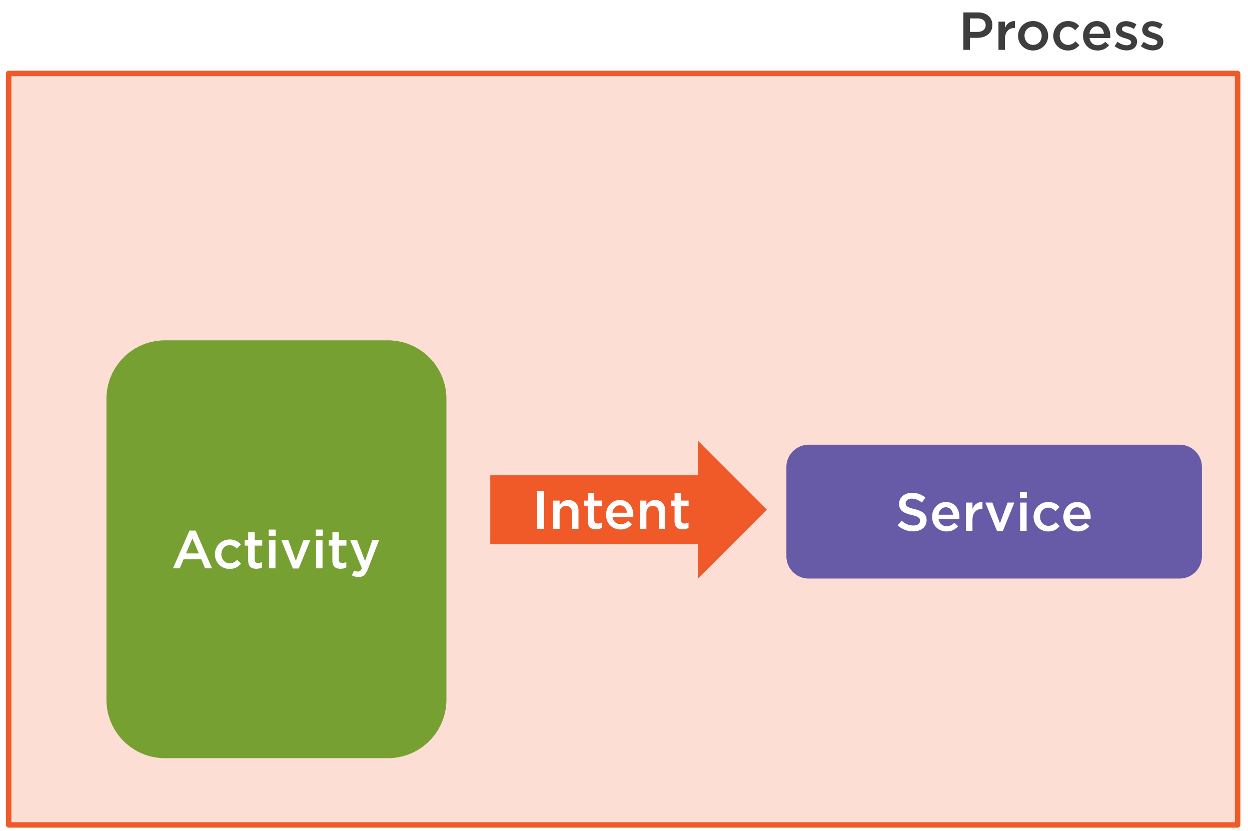
Services are started similar to activities

- Create an intent
- Associate extras with intent
- Intent passed to Context.startXXX
- Offer basically the same startup options

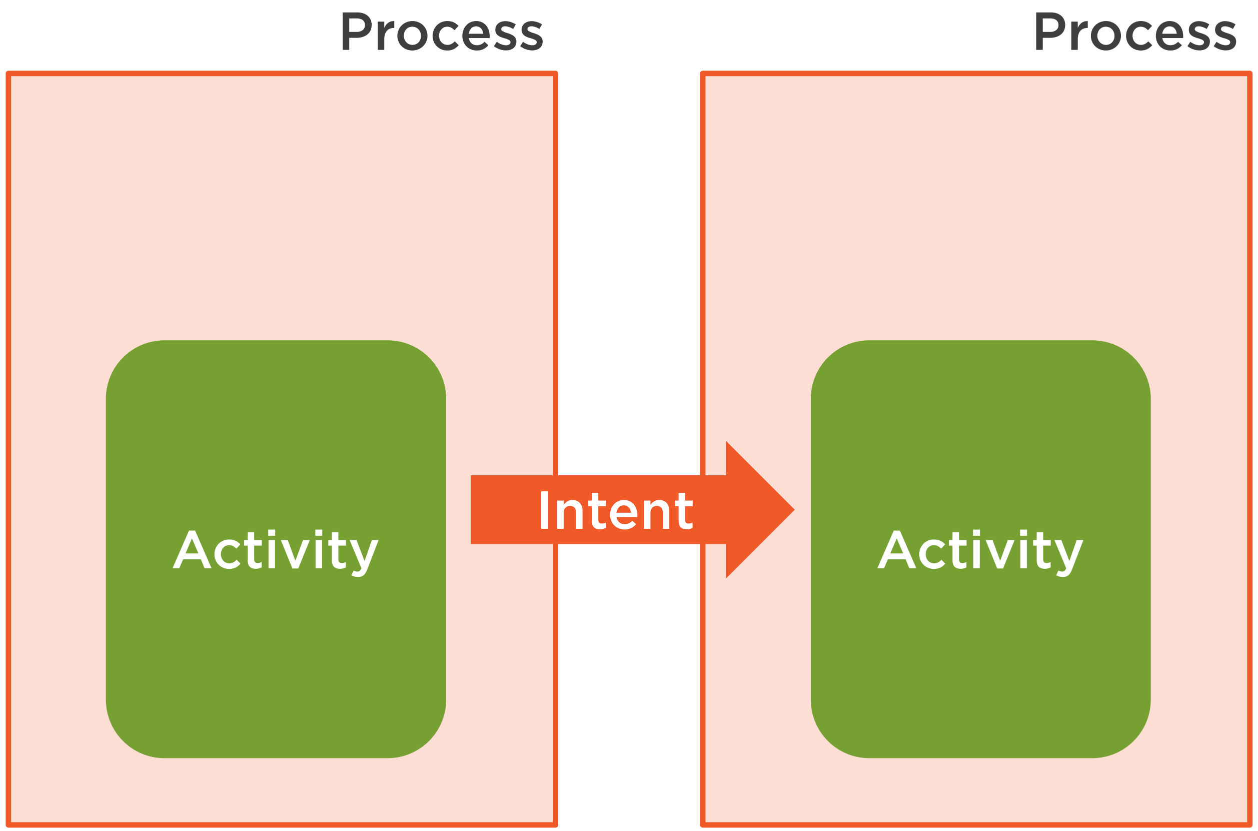
Starting from Same App



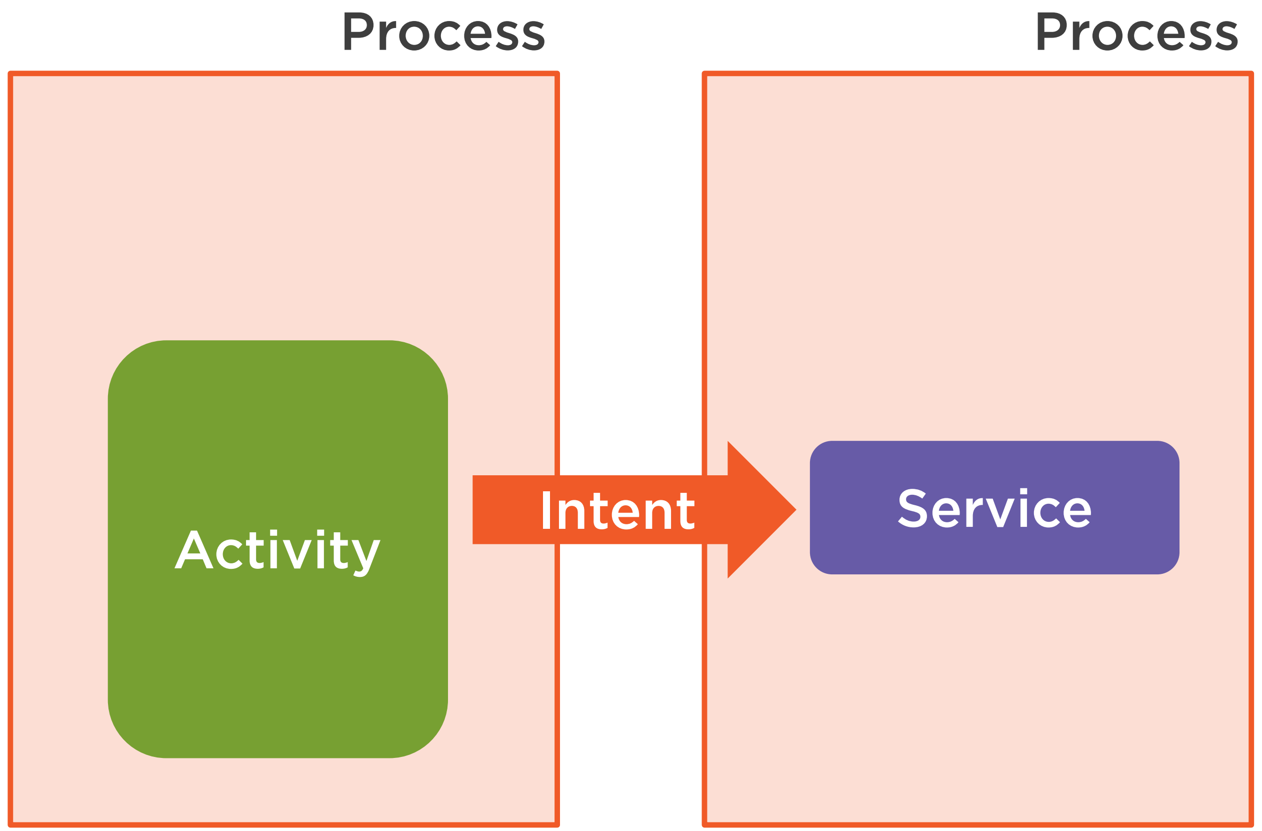
Starting Within a Process



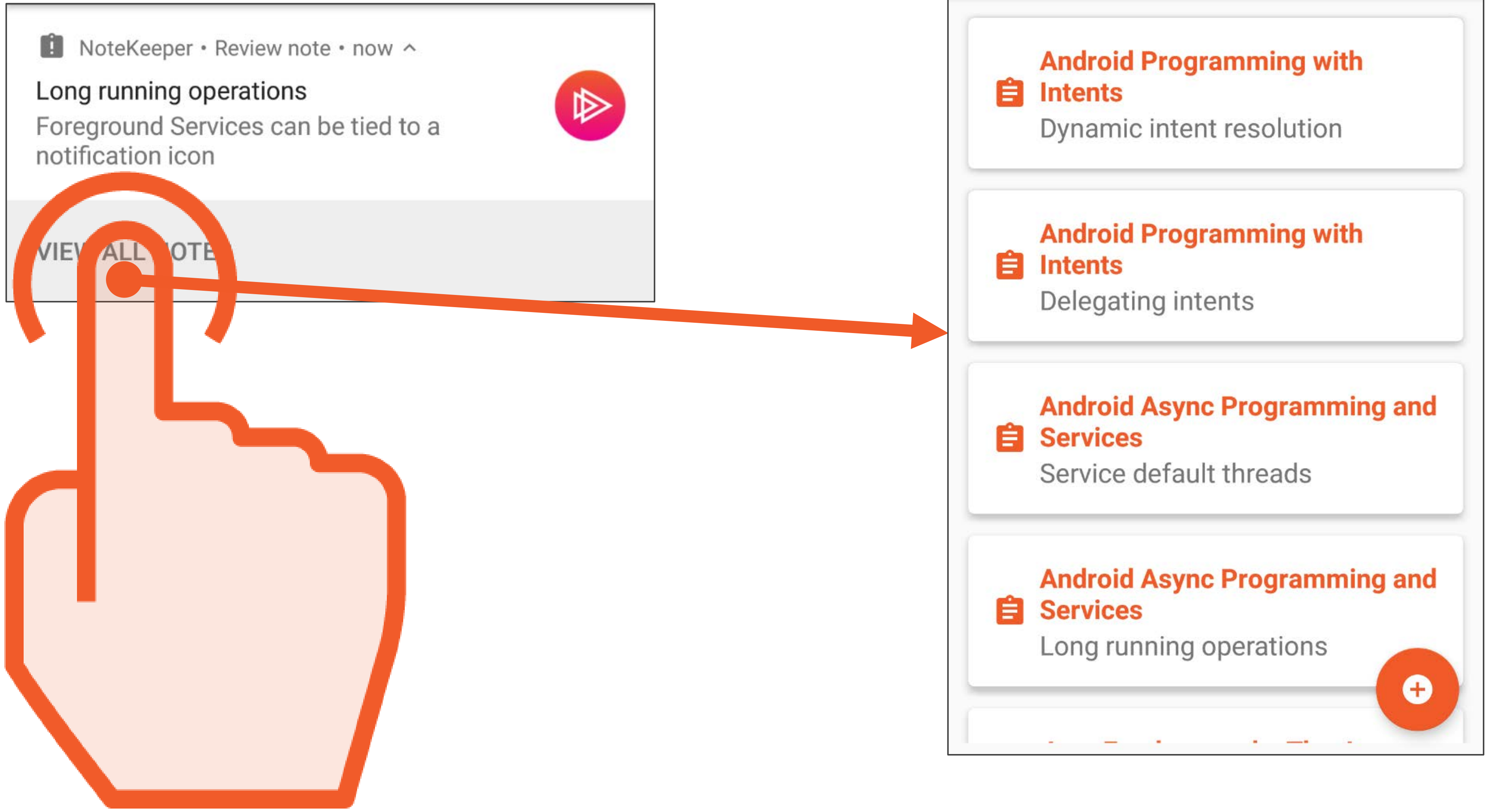
Starting from Another App



Starting from Another App



Starting with a PendingIntent


NoteKeeper • Review note • now

Long running operations

Foreground Services can be tied to a notification icon

VIEW ALL NOTES

NoteKeeper

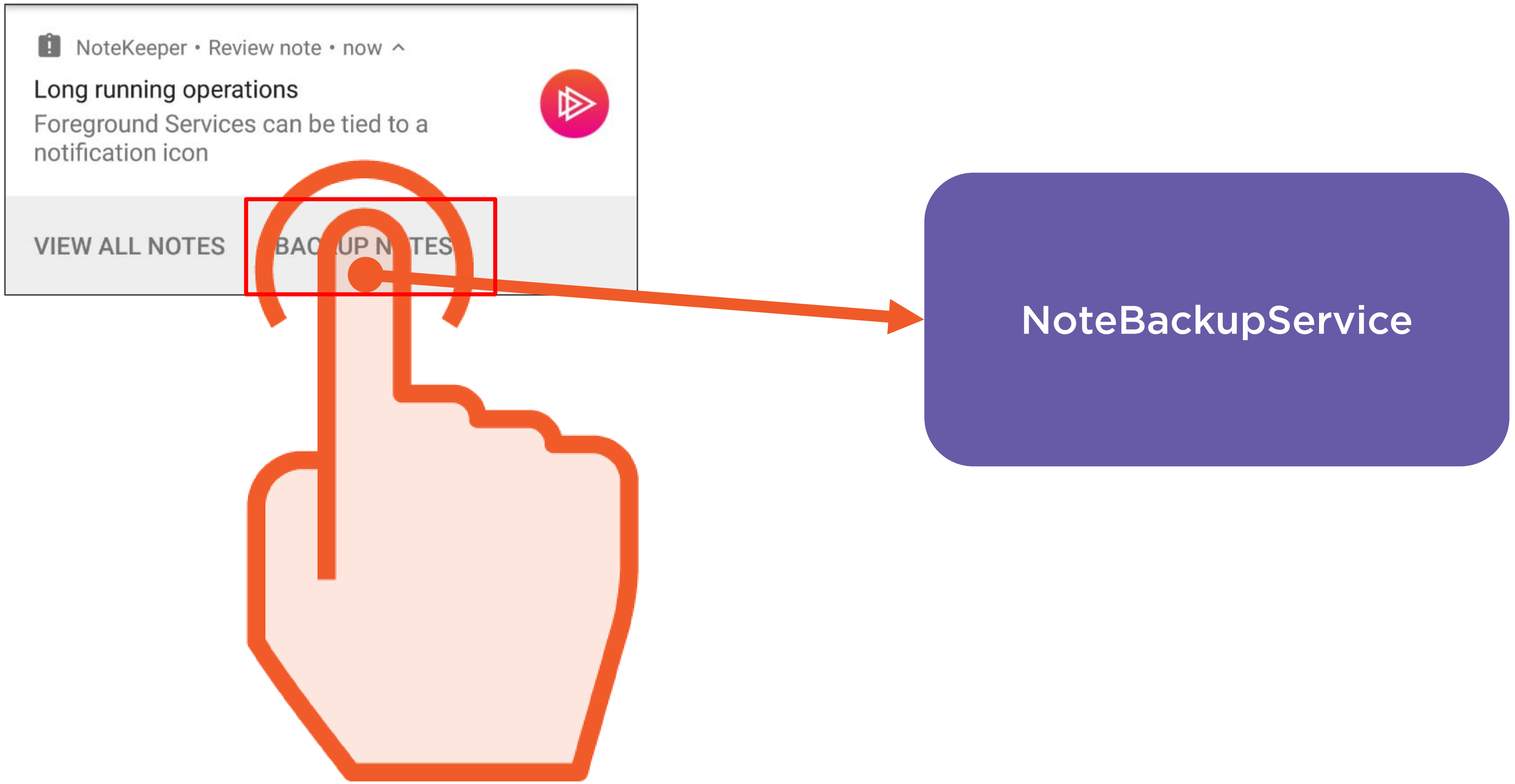
Android Programming with Intents
Dynamic intent resolution

Android Programming with Intents
Delegating intents

Android Async Programming and Services
Service default threads

Android Async Programming and Services
Long running operations

Starting with a PendingIntent



Long running operations

Foreground Services can be tied to a notification icon

Starting a Service

Starting from another app

- Service must be marked as exported in application manifest

Starting with PendingIntent

- Create the PendingIntent instance with PendingIntent.getService

Summary

Activities can initiate background work

- OK for work of a few seconds or less
- Longer work at risk of being terminated if user switches away from the app

Services perform non-UI work

- Reliably perform long-running work
- Continue running even if user switches to another app

Summary

Services extend the Service class

- Services are very flexible
- Directly extending Service class requires handling housekeeping details

Commonly extend the IntentService class

- Simplifies service implementation
- Works well for most common scenarios

Summary

Starting a service

- Create service intent along with extras
- Pass intent to Context.startService

Can associate service with PendingIntent

- Create service intent along with extras
- Use PendingIntent.getService