UNIVERSITY of WASHINGTON

Android 210 - Lecture 8
Threading, AsyncTask, IntentService

Margaret Maynard-Reid
March 9, 2015

Course Evalution

Take 15 minutes to provide feedback

Agenda

- Homework 4 Solution
- Threading
- AsyncTask
- IntentService

Sample Code

- SampleAsyncTask
- SampleIntentService

Android Stories

- Samsung Pay vs. Apple Pay: There's a difference
- Google’s Android to Take On Facebook in Virtual Reality
- 18 smartphones announced at Mobile World Congress 2015

Homework 4 Solution

Walk through homework 4 solution

Lessons learned?

Review from last week

- What is an Intent?
- What is an Explicit intent?
- What is an Implicit intent?
- An example of how IntentFilter is used?
- What is a BroadcastReceiver?
- An example of Android system broadcast?
- How do you register a BroadcastReceiver?

Break

Process

Concurrent instance of Application execution

- Scheduled by the system
- Created via
  - Android Launcher
  - ProcessBuilder
- Usually all components run within the same Process
  - Share the same memory
  - This can be changed/merged with android:process attribute
  - Applications must share a user ID and signed certificate

Process Hierarchy

Process Hierarchy – Android's assessment of the importance of a Process

- Foreground
- Visible
- Service
- Background
- Empty

http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html

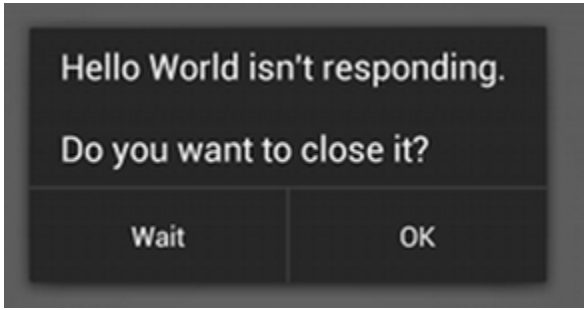
Thread

Concurrent instance of Process execution

- Scheduled by the system
- Can be created via
  - Thread
  - AsyncTask

ANR (Application Not Responding)

- Not respond to UI input event within 5 seconds.
- BroadcastReceiver didn't finish executing within 10 seconds.



http://developer.android.com/training/articles/perf-anr.html

Android Main Thread

By default,

- Android starts a new process for the app with a single thread
- All components of the same app run in the same process and thread -
- The main thread, also called the UI thread
- You can’t update UI from another thread

A Responsive UI

1. Don’t block the UI! Move long running operations off of the UI thread:
   - loading images
   - loading large files
   - accessing databases
   - fetching data via network
2. Provide UI indication on progress
3. Update UI once operation is done

Update UI - the challenge

Android doesn't allow you to access UI from the background. A couple of solutions:

- Activity.runOnUiThread(Runnable)
- View.post(Runnable)
- Handler() framework
- AsyncTask
- Receiver (i.e. IntentService)

Handler framework

- Part of the Android framework for handling threads
- You can create a handler on the UI thread then use it to communicate from background to UI

https://developer.android.com/training/multiple-threads/communicate-ui.html

Handler Pros & Cons

Pros

- More control
- Generic thread communication
- Not tied to activity or UI

Cons

- Complex

AsyncTask

Wrapper around a Thread and Handler

- Useful for getting threaded operations off of the main UI thread
- Pretty commonly used in Android
- Suitable for “short” operations (a few seconds)

AsyncTask

- Generic types used for parameters
  - Params - any parameters coming into the AsyncTask when executed
  - Progress - progress value reported via onProgressUpdate
  - Result - any parameters coming from the completion of the AsyncTask
- If nothing is being used or passed, use the Void generic type

AsyncTask

Method	When/What	UI Thread?
onPreExecute() -optional	Run before doInBackground() is executed Does setup if needed	Yes
doInBackground() **-Required**	**Executes your code on a background thread**	**No**
onProgressUpdate() -optional	Takes Progress parameter values from publishProgress() calls	Yes
onPostExecute() -optional	Takes the Result parameter value after doInBackground() finishes Does UI updates	Yes

Creating an AsyncTask

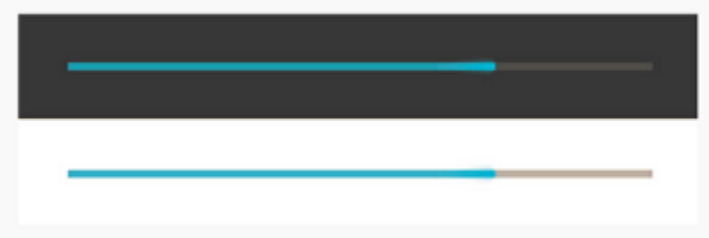
- Extend the AsyncTask base class
- Implement doInBackground() at a minimum
- Use the other methods as needed

AsyncTask Example

```
private class MyAsyncTask extends AsyncTask<Void, Void, Void> {
@Override
protected String doInBackground(Void... params) {
        while (true) {
        Log.i(LOG_TAG, "I'm running in the background");
        }
    }
}
```

ProgressBar

- Keep user informed of progress (of the background operation)
- Dismiss it after operation is done



Hands-on

- Walk through SampleAsyncTask

Break

AsyncTask

- Pros
  - Fairly easy to create with the boilerplate template
  - No need to worry about creating/managing your own thread
  - Allows you to easily update UI
- Cons
  - Can only handle operations of a few seconds
  - Tied to the activity that calls it

When to use a Service

- You can get a lot done with an AsyncTask
- If you need logic, system interaction, or resilience when executing a background action, consider using a Service.

What are some other specific examples?

Service

- A Service is a component of your application process that provides background processing
- Defined in the Manifest via <service/>
- When started or bound the system instantiates the Service and calls its onCreate()
  - Usually runs in the application process (by default)
  - It runs on the main thread

Service

This is important:

– By default, a Service runs on your main thread

Examples of why this is important?

<service>

Manifest entry for <service>

- Can have zero or more IntentFilters
- android:process - allows you to run your service in another process

IntentService

- A base class for Service
- Offloads tasks from UI thread
- All requests handled in a single worker thread
- Automatically stops when all requests are processed

IntentService

1. Create Intent Service,
2. Make sure there is an empty constructor calling super
3. Update onHandleIntent()
4. Register service in AndroidManifest.xml
5. Create a BroadcastReceiver
6. Register the Receiver in the activity to update UI

Hands-on

- Walk through SampleIntentService

IntentService Pros & Cons

- Pros
  - Fairly simple service
  - Automatically shuts down
- Cons
  - Cannot easily interact with UI
  - Only one request is processed at a time, sequentially
  - Operations running in IntentService can’t be interrupted

Next week

- 3/15 Project presentation and source code due on Catalyst Dropbox
- 3/16 Project presentation!