Android & Java Concurrency: the Half-Sync/Half-Async Pattern (Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Module

• Understand how *Half-Sync/Half-Async* is implemented & applied in Android
Half-Sync/Half-Async POSA2 Concurrency

Implementation

- Decompose overall system into three layers: synchronous, asynchronous, & queueing

```java
public abstract class AsyncTask<Params, Progress, Result> {
    public final AsyncTask<Params, Progress, Result>
        execute(Params... params) {
        return executeOnExecutor(sDefaultExecutor, params);
    }

    public final AsyncTask<Params, Progress, Result>
        executeOnExecutor(Executor exec, Params... params) {
        onPreExecute();
        mWorker.mParams = params;
        exec.execute(mFuture);
        return this;
    }
    ...
}
```

Identify short-duration services & implement them in the async layer

`frameworks/base/core/java/android/os/AsyncTask.java` has the source code
Implementation

- Decompose overall system into three layers: synchronous, asynchronous, & queueing

```java
public abstract class AsyncTask<Params, Progress, Result> {
    public AsyncTask() {
        mWorker = new WorkerRunnable<Params, Result>() {
            public Result call() throws Exception {
                ... return postResult(doInBackground(mParams));
            }
        };
    }
    ...
```

Identify long-duration services & implement them in the sync layer in a background thread
Implementation

- Decompose overall system into three layers: synchronous, asynchronous, & queueing

```java
public class ThreadPoolExecutor
    extends AbstractExecutorService {
    /**
     * The queue used for holding tasks and handing off to worker
     * threads. */
    private final BlockingQueue<Runnable> workQueue;
```
Implementation

- Decompose overall system into three layers: synchronous, asynchronous, & queueing
- Implement the services in the synchronous layer

```java
class DownloadAsyncTask extends 
    AsyncTask<String, Integer, Bitmap> {
    ...
    protected Bitmap 
    doInBackground(String... url) {
        return downloadImage(url[0]);
    }
}
```

Download in background thread
Implementation

- Decompose overall system into three layers: synchronous, asynchronous, & queueing
- Implement the services in the synchronous layer
- Implement the services in the asynchronous layer

```java
class DownloadAsyncTask extends AsyncTask<String, Integer, Bitmap> {
    protected void onPreExecute() {
        dialog.display();
    }

    protected void onPostExecute(Bitmap bitmap) {
        performPostDownloadOperations(bitmap);
        dialog.dismiss();
    }
}
```

Perform on UI thread
Implementation

• Decompose overall system into three layers: synchronous, asynchronous, & queueing
• Implement the services in the synchronous layer
• Implement the services in the asynchronous layer
• Implement (or reuse) the queueing layer

```java
public class ThreadPoolExecutor extends AbstractExecutorService {
    ...
    private Runnable getTask() {
        ...
        Runnable r = workQueue.take();
        ...
        return r;
        ...

    public void execute(Runnable command) {
        ...
        workQueue.offer(command);
        ...
```

Applying Half-Sync/Half-Async in Android
Applying Half-Sync/Half-Async in Android

UI thread receives user request to download a particular URL & calls execute()
Applying Half-Sync/Half-Async in Android

- Download Activity
- DownloadAsyncTask
- Executor
- BlockingQueue
- WorkerRunnable

The UI thread performs short-duration operations

`execute(url)`

`onPreExecute()`

`execute()`

`onPostExecute()`

`setDefaultExecutor(AsyncTask.THREAD_POOL_EXECUTOR)`

`call()`

`postResult()`

`offer()`

`take()`

`doInBackground()`
Applying Half-Sync/Half-Async in Android

**Executor** runs a pool of worker threads that remove user request, downloads file synchronously, & then displays the result on the UI Thread.
Applying Half-Sync/Half-Async in Android

- **Download Activity**: execute(url)
- **Download AsyncTask**: onPreExecute(), doInBackground(), onPostExecute()
- **Executor**: setDefaultExecutor(AsyncTask.THREAD_POOL_EXECUTOR)
- **Blocking Queue**: offer(), take(), call()
- **Worker Runnable**: doInBackground(), onPostExecute()

A synchronized request queue mediates access between the sync & async layers
Applying Half-Sync/Half-Async in Android

- Download Activity
- Download AsyncTask
- : Executor
- : Blocking Queue
- : Worker Runnable

If flow control occurs on a connection each thread can block without degrading the QoS of other threads in the pool
Applying Half-Sync/Half-Async in Android

- **Download Activity**
  - `execute(url)`
  - `onPreExecute()`
  - `onPostExecute()`

- **Download AsyncTask**
  - `execute()`
  - `doInBackground()`
  - `UI Thread` performs short-duration operations

- **Executor** (AsyncTask.THREAD_POOL_EXECUTOR)
  - `setDefaultExecutor()`
  - `call()`
  - `postResult()`

- **Blocking Queue**
  - `offer()`
  - `take()`

- **Worker Runnable**
  - `Worker Threads`
Summary
Summary

- The Android AsyncTask framework implements *Half-Sync/Half-Async* pattern to encapsulate the creation of background thread processing & synchronization with the UI Thread.
- It also supports reporting progress of the running tasks.

*AsyncTask framework*

Synchronous Service Layer

Queueing Layer

Asynchronous Service Layer

- **Background Thread**
  - Thread 1
  - Thread 2
  - Thread 3

- **BlockingQueue**

- **MyActivity**

- **UI Thread Looper**

*Downloading Video ...*
**Summary**

- The Android AsyncTask framework implements *Half-Sync/Half-Async* pattern to encapsulate the creation of background thread processing & synchronization with the UI Thread.
- AsyncTask is a sophisticated implementation of the *Half-Sync/Half-Async* pattern.
  - e.g., there are multiple interactions between the sync & async portions via various queues.
End of Android & Java Concurrency: The Half-Sync/Half-Async Pattern (Part 2)