Chapter 15

Dealing with Threads

Ideally, you want your activities to be downright snappy, so your users don’t feel that your application is sluggish. Responding to user input quickly (e.g., within in 200 milliseconds) is a fine goal. At minimum, though, you need to make sure you respond within 5 seconds, lest the ActivityManager decide to play the role of the Grim Reaper and kill off your activity as being nonresponsive.

Of course, your activity might have real work to do, which takes nonnegligible amount of time. This invariably involves the use of a background thread. Android provides a veritable cornucopia of means to set up background threads, yet allow them to safely interact with the UI on the UI thread.

The “safely interact” concept is crucial. You cannot modify any part of the UI from a background thread. That must be done on the UI thread. This generally means that there will need to be some coordination between background threads doing the work and the UI thread showing the results of that work.

This chapter covers how to work with background and UI threads in your Android applications.

Getting Through the Handlers

The most flexible means of making an Android-friendly background thread is to create an instance of a Handler subclass. You need only one Handler object per activity, and you do not need to manually register it. Merely creating the instance is sufficient to register it with the Android threading subsystem.

Your background thread can communicate with the Handler, which will do all of its work on the activity’s UI thread. This is important, as UI changes, such as updating widgets, should occur only on the activity’s UI thread.

You have two options for communicating with the Handler: messages and Runnable objects.
Messages

To send a Message to a Handler, first invoke `obtainMessage()` to get the Message object out of the pool. There are a few flavors of `obtainMessage()`, allowing you to create empty Message objects or ones populated with message identifiers and arguments. The more complicated your Handler processing needs to be, the more likely it is you will need to put data into the Message to help the Handler distinguish different events.

Then you send the Message to the Handler via its message queue, using one of the `sendMessage...()` family of methods, such as the following:

- `sendMessage()`: Puts the message on the queue immediately.
- `sendMessageAtFrontOfQueue()`: Puts the message on the queue immediately, placing it at the front of the message queue, so your message takes priority over all others.
- `sendMessageAtTime()`: Puts the message on the queue at the stated time, expressed in the form of milliseconds based on system uptime (`SystemClock.uptimeMillis()`).
- `sendMessageDelayed()`: Puts the message on the queue after a delay, expressed in milliseconds.

To process these messages, your Handler needs to implement `handleMessage()`, which will be called with each message that appears on the message queue. There, the handler can update the UI as needed. However, it should still do that work quickly, as other UI work is suspended until the Handler is finished.

For example, let's create a ProgressBar and update it via a Handler. Here is the layout from the Threads/Handler sample project:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ProgressBar android:id="@+id/progress"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

The ProgressBar, in addition to setting the width and height as normal, also employs the style property. This particular style indicates the ProgressBar should be drawn as the traditional horizontal bar showing the amount of work that has been completed.

And here is the Java:

```java
package com.commonsware.android.threads;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
```