CPET 565/CPET 499 Mobile Computing Systems
Lecture Note 9
Sept. 29, 2014

The Android Developer's Cookbook, 2nd edition

- **Chapter 3 Threads, Handlers, Alerts, Services, and Broadcast Receivers**

Threads

- Every application by default runs a single **process** upon creation that contains all the tasks. To avoid hanging the UI, time consuming tasks, such as network downloads or computationally intensive calculations, should reside in a background thread.
- Class Thread extends Object Implements Runnable, http://developer.android.com/reference/java/lang/Thread.html
    - A Thread is a concurrent unit of execution.
    - It has its own call stack for methods being invoked, their arguments and local variables.
    - Each application has at least one thread running when it is started, the main thread, in the main ThreadGroup.
    - The runtime keeps its own threads in the system thread group.
- Methods
    - Start a thread: start()
    - Put a thread to sleep: sleep()
    - Pause/Rusume a thread: onPause(); onResume; paused = false, paused = true … FLAG
    - Thread priority:
        - Thread.MIN_PRIORITY == 1; Thread.MAX_PRIORITY == 10;
        - Android.os.Process.setThreadPriority()
    - Cancelling/Killing a Thread:
        - Stop() .. deprecated because it might leave the application in a unpredictable state
        - interrupt() method
        - setDaemon(true) … Declare all spawned thread as daemon thread and ensure that threads associated with the application are killed when the application's main thread is killed.
        - PUT while (stillRunning) loop in the run() method and externally stillRunning = false to kill the thread.

Handlers

- A main thread (time-critical thread) ⇔ background thread (time consuming thread)
- Handlers
    - Objects for sending messages between threads
    - Each handler is bound to a single thread, delivering message to it and executing commands from it
- Class Handler extends Object, http://developer.android.com/reference/android/os/Handler.html
    - A Handler allows you to send and process Messages and Runnable Objects associated with a thread's MessageQueue
    - Known Direct Subclasses:

        - AsyncQueryHandler
        - AsyncQueryHandler.WorkerHandler
        - HttpAuthHandler
        - SslErrorHandler

Alerts

- Provides a quick message to the user outside the application's main UI.
- It can be in an overlay window: such as "a toast alert" or "AlertDialog" box
    - Toaster alert: a printed message to the screen with a single line of code; equivalent to printf() in C programs; can be used as a debug tool
- It can also be in the "notification bar" at the top of the screen
- Class AletDialog extends Dialog implements DialogInterface, http://developer.android.com/reference/android/app/AlertDialog.html

Services

- Class Services extends ContextWarpper implements ComponentCallbacks, http://developer.android.com/reference/android/app/Service.html
- A service is an Android component that runs in the background, do short-lived tasks with a low-priority, without user interaction.
    - It can be started and stopped by an components
- Foreground service is supported, but this requires setting a mandatory ongoing notification in the "notification bar" so the user is informed about a service taking priority.
- IntentService class.. is a service that holds a "QUEUE" of intent it has received and executed them one by one.
    - This is an ideal worker thread for many background tasks likes
        - Polling servers for new information or downloading large amount of data
- Examples
    - Activity => UI for select music files => start a service to play back the files
    - Activity => UI to upload a set of picture to a website
    - A broadcast receiver receives a message that a picture was taken and launches a service to upload the new picture to a website.

Broadcast Receivers

- Class BroadcastReceiver extends Object, http://developer.android.com/reference/android/content/BroadcastReceiver.html
- A broadcast receiver listens for relevant broadcast messages to trigger an event.
- Some examples of broadcast events already sent from the OS are
    - The camera button was pressed.
    - The battery is low.
    - A new application was installed.
- A user-generated component can also send a broadcast, such as
    - A calculation was finished.
    - A particular thread has started.

Chapter 3: Developer Cook Book Examples – Recipes

Threads

- **Recipe: Launching a Second Thread: Listing 3-2, pp. 54-55**

- Recipe: Creating a Runnable Activity, Listing 3-3, pp. 55-56
- Recipe: Setting a Thread's Priority, pp. 56
- Recipe: Cancelling a Thread, pp. 57
- Recipe: Sharing a Thread Between Two Applications, pp. 57-58

<u>Messages between Threads: Handlers</u>

- A main thread (time-critical thread) ⇔ background thread (time consuming thread)
- Handlers
    - Objects for sending messages between threads
    - Each handler is bound to a single thread, delivering message to it and executing commands from it
- Recipe: Scheduling a Runnable Task from the Main Thread; Listings 3.4, 3.5, pp. 58-60
    - Timer … running as a background thread so it does not block the UI thread
    - UI thread … needs update whenever the time changes
    - The handler mHandler is created and used to QUEUE the runnable object mUpdateTimeTask
      ```
      private Handler mHandler = new Handler();
      if (mStartTime == 0L){…
      mHandler.remov
      ```
    - Perform recursive call in the task itself continues to update the time every 200 ms.
      ```
      mHandler.postDelayed(this, 200)
      ```
- Recipe: Using a Countdown Timer; Listing 3.6, pp. 60-61
  ```
  import android.os.CountDownTimer
  …
  new CountDownTimer (3000,1000); // two arguments: total time duration,  or time interval to process onTick()
  onTick() method
  ```
- Recipe: Handling a Time-Consuming Initialization; Listings 3.7, 3.8, pp. 61-63
    - Loading splash screen specified in: res/layout/loading.xml
      ```
      <LinearLayout …
                  <TextView …
                   Android:text = "Loading…"
      </LinearLayout>
      ```
    - initializaArrays() … running in background
      ```
      public class HandleMessage extends Activity implements Runnable{ …
                  //
                  Thread thread = new Thread(this);
                  Thread.start();
      ```

<u>Alerts</u>

- Recipe: Using Toast to Show a Brief Message on the Screen, pp. 63-64
- Recipe: Using an AlertDialog Box, Listing 3.9, pp. 64-65
    - AlertDialog class
- Recipe: Showing Notification in the Status Bar; Listings 3.10, 3.11, and 3.12, pp. 65-69

<u>Services</u>

- Recipe: Creating a Self-Contained Service (with a single components); Listings 3.13, 1.14, 3.15, pp. 70-74
    - Service Lifecylce: http://developer.android.com

    - AndroidManifest.xml
        - <service android:name=”myService”></service>
    - Override onCreate(), onDestroy()
    - Override onBind()
    - startService(), stopService()
- Recipe: Adding a WakeLock; Listing 3.17, 3.18, pp. 74-77
    - WakeLock Type: (CPU, Screen, Hardware)
        - PARTICL_WAKE_LOCK: on, off, off
        - SCREEN_DIM_WAKE_LOCK: on, dimmed, off
        - SCREEN_BRIGHT_WAKE_LOCK: on, bright, off
        - FULL_WAKE_LOCK: on, bright, bright, bright
    - PowerManager class
    - Context.getSystemService(Context, PowerService)
    - Create a new WakeLock instance
        - powerManager.mWakeLock = powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, LOG_TAG)
    - setWakeLock()
    - releaseWakeLock()
    - To activate WakeLOck … mWakeLock.acquire()
    - To release WakeLocks
        - mWakeLock.isHeld()
        - mWakeLock.release()
- Recipe: Using a Foreground Service, Listing 3.19, pp. 77-79
    - Activating the foreground service:
        - onStart()
        - startForeground(NOTIFICATION_ID, getForegroundNottification());
    - Stopping foreground service:
        - onDestroy()
        - stopForeground(true)
            - to remove notification
- Recipe: Using an IntentService, Listing 3.20, 3.21, and 3.22
    - intentService class
    - IntentQueue … EMPTY/FULL
    - handleIntent()
    - intent.getStringExtra(“msg”);

Broadcast Receivers

- Recipe: Starting a Service when the Camera Button is Pressed; Listings 3.23, 3.24, 3.25, pp. 83-85