The Internet and Threading

A stitch in time saves 9… errors

The Internet

First off:

- Put this in your manifest:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Basic Connection

```
HttpURLConnection conn = (HttpURLConnection)
      new URL(“http://mtmurdock.com”)
      .openConnection();
InputStream is = conn.getInputStream();
```

InputStream

```
BufferedInputStream bis = new BufferedInputStream(in);
ByteArrayOutputStream buf = new ByteArrayOutputStream();
int result = bis.read();
while(result != -1) {
   byte b = (byte)result;
   buf.write(b);
   result = bis.read();
}
String s = buf.toString();
```

Better ways

```
// use scanner trick
String s = new Scanner(is)
      .useDelimiter("\\A").next();

// use CommonsIO (external library)
StringWriter writer = new StringWriter();
IOUtils.copy(is, writer);
String s = writer.toString();
```

Headers

```
// adds or sets value for key
conn.setRequestProperty(“key”, “value”);

// adds value if doesn’t exist
conn.addRequestProperty(“key”, “value”);
```

Method

```
// “GET”, “POST”, “PUT”, “DELETE”, or anything
conn.setRequestMethod(“GET”);
```

Query Parameters

- Parameters passed to server
- Similar to parameters on method call
- Passed differently based on method

Query Parameters

```
String query = “key1=value1&key2=value2&key3=value3”;

// GET
HttpURLConnection conn = (HttpURLConection)
      new URL(url + “?” + query).openConnection();
InputStream is = conn.getInputStream();

// POST
HttpURLConnection conn = (HttpURLConection)
      new URL(url).openConnection();
conn.setRequestMethod(“POST”);
OutputStream os = connection.getOutputStream();
os.write(query.getBytes());
InputStream is = conn.getInputStream();
```

Advanced Connections

- If this isn't cutting it, you can use the apache http classes.
- HttpClient, HttpServer, HttpRequest, HttpResponse, etc.
- Google suggests using URLConnection

Threading

The Main Thread

- Also called the GUI thread.
- Primary thread of execution.
- onCreate(), onMeasure(), onPause(), onDraw(), and basically all ‘on’ methods.

Rules

1. NEVER PERFORM BLOCKING OPERATIONS ON THE MAIN THREAD
2. NEVER ACCESS THE GUI FROM ANY THREAD OTHER THAN THE MAIN THREAD

Access Main Thread

- Activity.runOnUiThread(Runnable)
- View.post(Runnable)
- View.postDelayed(Runnable, millis)

Thread

- Basic threading class

```
Thread t = new Thread(new Runnable() {
      public void run() {
            // thread code here
      }
});
t.start();
```

Timer and TimerTask

- Thread is not well suited for repeating tasks.
- Inefficient, inaccurate for precise timing.

```
Timer timer = new Timer();
timer.schedule(new TimerTask() {
      // repeating code here
}, 0, 200);
```

AsyncTask

- Thread and Handler combined
- Allows you to run a background task and update the user interface along the way.
- Not a general purpose threading solution.

Implementing AsyncTask

```
public class MyTask extends
      AsyncTask<Param, Progress, Result>
{
}
```

- Param: input type
- Progress: update type
- Result: output type

Methods

- `onPreExecute` (main thread) called before doInBackground.
- `doInBackground` (background) main worker thread. Where the magic happens.
- `onProgressUpdate` (main thread) called ever time updateProgress is called by doInBackground.
- `onPostExecute` (main thread) called after doInBackground returns.

Canceling AsyncTask

- Check isCancelled() regularly in doInBackground(). Return immediately if true.
- Override onCancel().
- Called instead of onPostExecute().

Running your task

```
URL url1, url2, url3;
url1 = new URL(“http://www.something.com”);
url2 = new URL(“http://www.42.net”);
url3 = new URL(“http://www.yourmom.com”);

MyTask task = new MyTask()
      .execute(url1, url2, url3);
```

Questions?