Notice that the only way to quit this Web server, once it's started, is to kill the task that launched it (typically by typing ctrl-C at the command line). Of course, if you're running this server under MacOS, your entire machine locks up while this server is running, and the only way to stop the server is to force-quit it, or drop into MacsBug.

**An Asynchronous (Background) Web Server**

Let's now modify the basic Web server to provide more friendly asynchronous behavior. This should allow it to run on platforms such as MacOS that do not yet provide preemptive multitasking.

We will modify the simple Web server so that it spawns a separate thread that deals with the HTTP connection itself. This separate thread will basically call the doIt method from within the run method of a Thread. Here again we recycle the HttpClientHandler class to handle a single http transaction after a client has connected.

**CAUTION**

While this version of httpd should, in theory, allow a series of clients to connect to a MacOS-based server running MacTCP, the reality is that when we tested this server with the current MacOS JDK, the server would sporadically quit with a fairly low-level monitor error being dumped to stderr. By the time this book is published, a new version of the MacOS Java runtime may be released that fixes this problem.

The source for the asynchronous Web server is shown below.

**Example 5-4  An Asynchronous (Background) Web Server Implementation**

```java
package JNC;

import java.io.*;
import java.net.*;
import java.util.*;

/**
 * Improved version of httpd server:
 * Places main loop into a separate thread,
 * so it doesn't lock up the system.
 */
public class HttpdAsync extends Thread {
    public final static int DEFAULT_PORT = 80;
```