This example presents a very simple HTTP daemon (httpd). In this example we just call ServerSocket.accept directly from our main thread. This blocks anything else from executing in our task. This presents a problem on platforms that have cooperative multitasking (such as MacOS), but not on platforms (Unix, Win32) which provide preemptive multitasking.

The task that called ServerSocket.accept (your code) blocks, but every other task in the system proceeds. On a platform that lacks preemptive multitasking, such as MacOS, this causes the entire system to hang waiting for a client to connect on the port. This is definitely not a well-behaved server and is useful only as an example.

The source code for this simplest Web server is shown below.

**Example 5-3 A Simple Web Server Implementation**

```java
package JNC;

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

/***
 * A class that implements a simple, synchronous HTTP server
 */
public class HttpdSimple
{
    public final static int DEFAULT_PORT = 80;

    public boolean fDebugOn = true;

    protected int fPort;
    private ServerSocket fMainListenSocket = null;
    private boolean fContinueListening = true;
    Socket fClientSocket = null;
    DataInputStream fClientInputStream;
    DataOutputStream fClientOutputStream;

    HttpTransactionHandler fTransactionHandler;

    /**
     * set the port
     */
    public HttpdSimple(int port) {
        if (port == 0) fPort = DEFAULT_PORT;
        else fPort = port;
    }  //HttpdSimple
```