Implements a multi-threaded Web proxy server

```
/*
C# Programming Tips & Techniques
by Charles Wright, Kris Jamsa

Publisher: Osborne/McGraw-Hill (December 28, 2001)
ISBN: 0072193794
*/

// Proxy.cs – Implements a multi-threaded Web proxy server
//
// Compile this program with the following command line:
// C:>csc Proxy.cs
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;
using System.Threading;

namespace nsProxyServer
{
public class ProxyServer
{
static public void Main (string [] args)
{
int Port = 3125;
if (args.Length > 0)
{
try
{
Port = Convert.ToInt32 (args[0]);
}
```

```
catch
{
Console.WriteLine (“Please enter a port number.”);
return;
}
}
try
{
// Create a listener for the proxy port
TcpListener sockServer = new TcpListener (Port);
sockServer.Start ();
while (true)
{
// Accept connections on the proxy port.
Socket socket = sockServer.AcceptSocket ();

// When AcceptSocket returns, it means there is a connection.
Create
// an instance of the proxy server class and start a thread
running.
clsProxyConnection proxy = new clsProxyConnection (socket);
Thread thrd = new Thread (new ThreadStart (proxy.Run));
thrd.Start ();
// While the thread is running, the main program thread will
loop around
// and listen for the next connection request.
}
}
catch (IOException e)
{
Console.WriteLine (e.Message);
}
}
}

class clsProxyConnection
{
```

```
public clsProxyConnection (Socket sockClient)
{
m_sockClient = sockClient;
}
Socket m_sockClient; //, m_sockServer;
Byte [] readBuf = new Byte [1024];
Byte [] buffer = null;
Encoding ASCII = Encoding.ASCII;

public void Run ()
{
string strFromClient = “”;
try
{
// Read the incoming text on the socket/
int bytes = ReadMessage (m_sockClient,
readBuf, ref strFromClient);
// If it's empty, it's an error, so just return.
// This will termiate the thread.
if (bytes == 0)
return;
// Get the URL for the connection. The client browser sends a
GET command
// followed by a space, then the URL, then and identifer for
the HTTP version.
// Extract the URL as the string betweeen the spaces.
int index1 = strFromClient.IndexOf (' ');
int index2 = strFromClient.IndexOf (' ', index1 + 1);
string strClientConnection =
strFromClient.Substring (index1 + 1, index2 – index1);

if ((index1 < 0) || (index2 < 0)) { throw (new IOException
()); } // Write a messsage that we are connecting.
Console.WriteLine ("Connecting to Site " +
strClientConnection); Console.WriteLine ("Connection from " +
m_sockClient.RemoteEndPoint); // Create a WebRequest object.
WebRequest req = (WebRequest) WebRequest.Create
```

(strClientConnection); // Get the response from the Web site. WebResponse response = req.GetResponse (); int BytesRead = 0; Byte [] Buffer = new Byte[32]; int BytesSent = 0; // Create a response stream object. Stream ResponseStream = response.GetResponseStream(); // Read the response into a buffer. BytesRead = ResponseStream.Read(Buffer,0,32); StringBuilder strResponse = new StringBuilder(""); while (BytesRead != 0) { // Pass the response back to the client strResponse.Append(Encoding.ASCII.GetString(Buffer, 0, BytesRead)); m_sockClient.Send(Buffer, BytesRead, 0); BytesSent += BytesRead; // Read the next part of the response BytesRead = ResponseStream.Read(Buffer, 0, 32); } } catch (FileNotFoundException e) { SendErrorPage (404, "File Not Found", e.Message); } catch (IOException e) { SendErrorPage (503, "Service not available", e.Message); } catch (Exception e) { SendErrorPage (404, "File Not Found", e.Message); Console.WriteLine (e.StackTrace); Console.WriteLine (e.Message); } finally { // Disconnect and close the socket. if (m_sockClient != null) { if (m_sockClient.Connected) { m_sockClient.Close (); } } } // Returning from this method will terminate the thread. } // Write an error response to the client. void SendErrorPage (int status, string strReason, string strText) { SendMessage (m_sockClient, "HTTP/1.0" + " " + status + " " + strReason + " "); SendMessage (m_sockClient, "Content-Type: text/plain" + " "); SendMessage (m_sockClient, "Proxy-Connection: close" + " "); SendMessage (m_sockClient, " "); SendMessage (m_sockClient, status + " " + strReason); SendMessage (m_sockClient, strText); } // Send a string to a socket. void SendMessage (Socket sock, string strMessage) { buffer = new Byte [strMessage.Length + 1]; int len = ASCII.GetBytes (strMessage.ToCharArray(), 0, strMessage.Length, buffer, 0); sock.Send (buffer, len, 0); } // Read a string from a socket. int ReadMessage (Socket sock, byte [] buf, ref string strMessage) { int iBytes = sock.Receive (buf, 1024, 0); strMessage = Encoding.ASCII.GetString (buf); return (iBytes); } } } [/csharp]