Assign-06 : My Own *Web Server*

Description:

This assignment can be done with a partner if you wish – or can be done alone. In this assignment, you will be investigating and coding *from the other side* – the server side. This assignment has you (and your partner) developing a simple, single threaded **web server**.

Objectives:

- To create a very simple, single user web server
- To better understand the parsing and creating of the underlying HTTP messages protocol used between a browser and a web server

Requirements:

1. The server must be written in C# as a console application. Your server has three mandatory command line arguments:
   - `-webRoot` : which will be set to the root folder for the website data (html, jpg and gif files)
   - `-webIP` : which will be set to the IP address of the server
   - `-webPort` : which will be set to the Port number the server will be listening on
2. Ensure that your server when compiled creates an executable called `myOwnWebServer`. So that when you invoke the server, you would enter a command like:
   ```
   myOwnWebServer -webRoot=C:\localWebSite -webIP=192.168.100.23 -webPort=5300
   ```
3. All incoming and outgoing functions supported by the server must comply with the HTTP/1.1 Protocol Specification.
4. The server is to be single threaded, so only 1 browser session needs to be supported at a time.
5. The server only supports the GET method in any incoming requests.
6. The server only supports the returned content types of text, HTML, jpg and gif.
7. The server must be able to parse the incoming HTTP request, and act accordingly. For example - If the request is for a text-based resource (e.g. `default.html`) and this file exists, then the server must be able to open the file, read the file contents and send its contents as a properly formatted HTTP response.
8. If an exception or error condition that arises in the server – you need to ensure that the proper HTTP status code is returned to the client as part of the response.
9. **You are not allowed to use** any helper classes (any of the `System.Net.Http*` set of classes) like those in the `System.Net` namespace. You must parse the incoming request, and formulate the full response header manually in your server code.
10. All files (html, jpg and gif) will be guaranteed of being in the server's root folder (or a sub-folder found within it).
11. Following best practices when creating any type of server-based application – ensure:
    - That the server outputs absolutely nothing to the console window it is running in
    - That the server generates a log file that tracks at least:
      1. i. The incoming HTTP Request header as well as the associated outgoing HTTP Response header (formatted so that it is easily readable)
      2. ii. Ensure that any other logging activity you have is also formatted to be readable

- That each log entry has a date and timestamp (e.g. `2015-11-01 14:05:00 <log entry goes here>`)

Note:

- There will be some challenges in this assignment – but if you (and your partner) properly design your server's solution, you will be able to satisfy the requirements.
- Feel free to use the HTTPTool (supplied in the course content) to exercise and debug your server, but don't forget to test your server with both the Internet Explorer v11 and Chrome browsers.
- Make sure that you get your MIME types correct in your response HTTP headers – check out this site for a complete list

Hand in:

1. ZIP up the final cleaned VS Solution directory structure and submit to the appropriate drop-box by the deadline.
    - Do not include your own test files – a set of pre-built test files will be used in the marking process.
2. Make sure to comment your classes, methods and code as per the SET Standard.
    - If you are working with a partner – ensure that your class header comments indicate the name of both partners