Linnéuniversitetet
Kalmar Växjö

Report

Assignment 2

1DV701

Author: Fabian Dacic, Yuyao Duan
Semester: Spring 2022
Email fd222fr@student.lnu.se, yd222br@student.lnu.se

Contents

Problem 1 – Implement a web server that accepts multiple client connections on a port

The first requirement of the program states that the user should be able to input a port and relative path to the "/public" directory. This requirement has been fulfilled through utilization of different checks (if statement – if the port number is negative or larger than the maximum port number; if there are at least two arguments inputted; if the user has inputted "/public" etc.)

To fulfill this requirement, pattern and matcher methods were utilized as well and if the user does not follow the instructions, it replaces whatever they had in the first place with the default values.

No third party libraries were used, and console output can be found throughout the different aspects of the program. The provided public folder has also been used and can be found together with the source file for the program. The program passed the python tests provided. The code can be found in the submission and the main program file is called WebServer.java.

There is also the matter of escaping the root and it was difficult to narrow down what was meant by it however we tried to mitigate the issue by validating the input before it is processed as it is evident in the main method (doing simple if checks and partly the reason why the pattern and the matcher are implemented), and throughout the other handlings of the requests too. It is essentially hardcoded regardless of whether the user puts the right input or not and any attempt to change it will be overwritten.

1.1 Discussion

Figure 1. Accessing index page

The index page was accessed using the following link in the browser: “localhost:[port chosen]/index.html”. Then the user is transported to the index page that states “Main index page” indicating success.

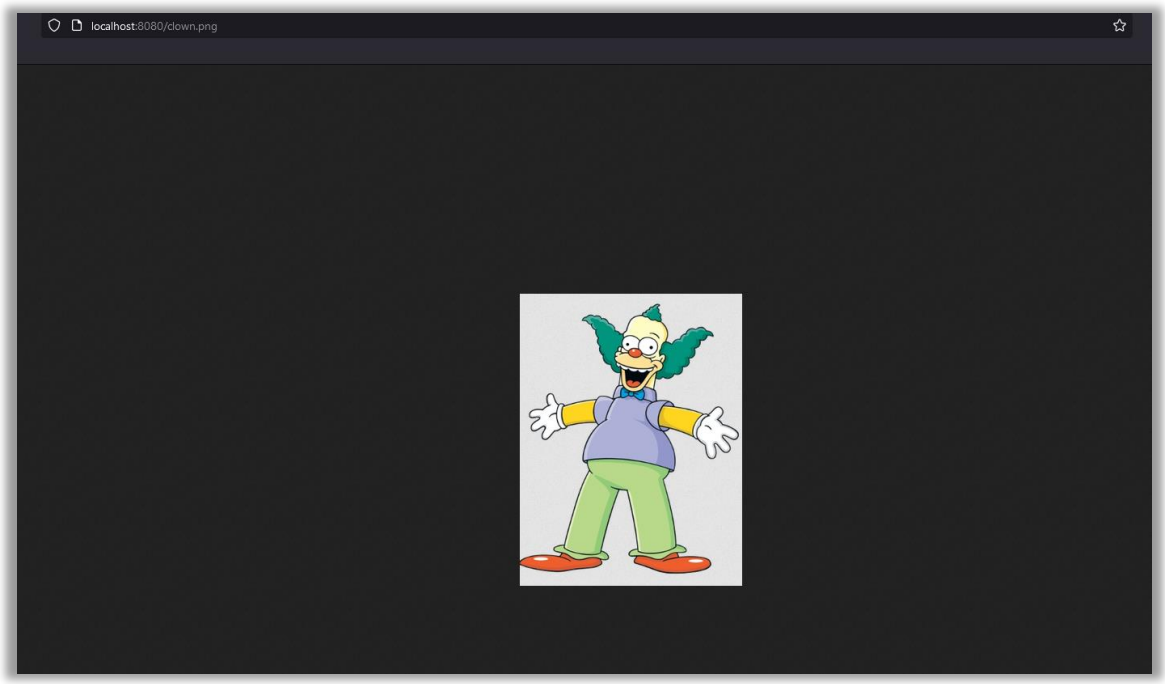

Figure 2. Accessing picture page

The index page was accessed using the following link in the browser: “localhost:[port chosen]/clown.png”. Then the user is transported to the page that shows a cartoon character indicating success.

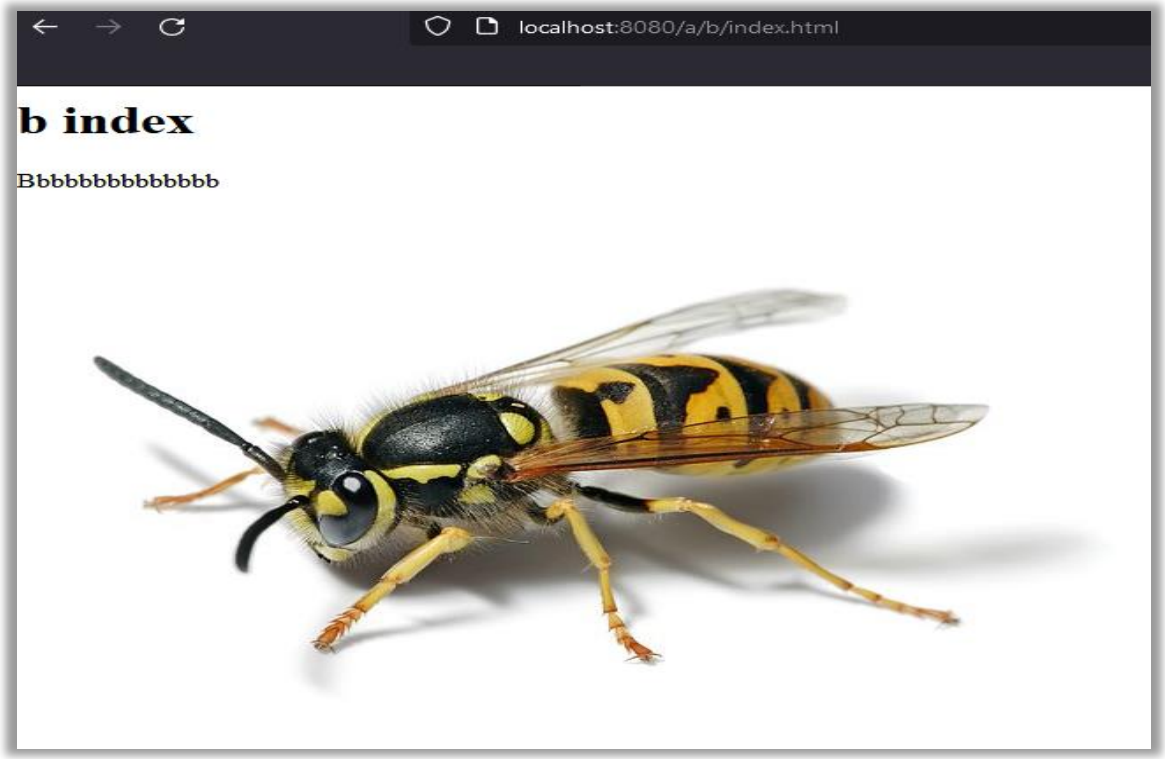

Figure 3. Accessing page within a few subdirectories

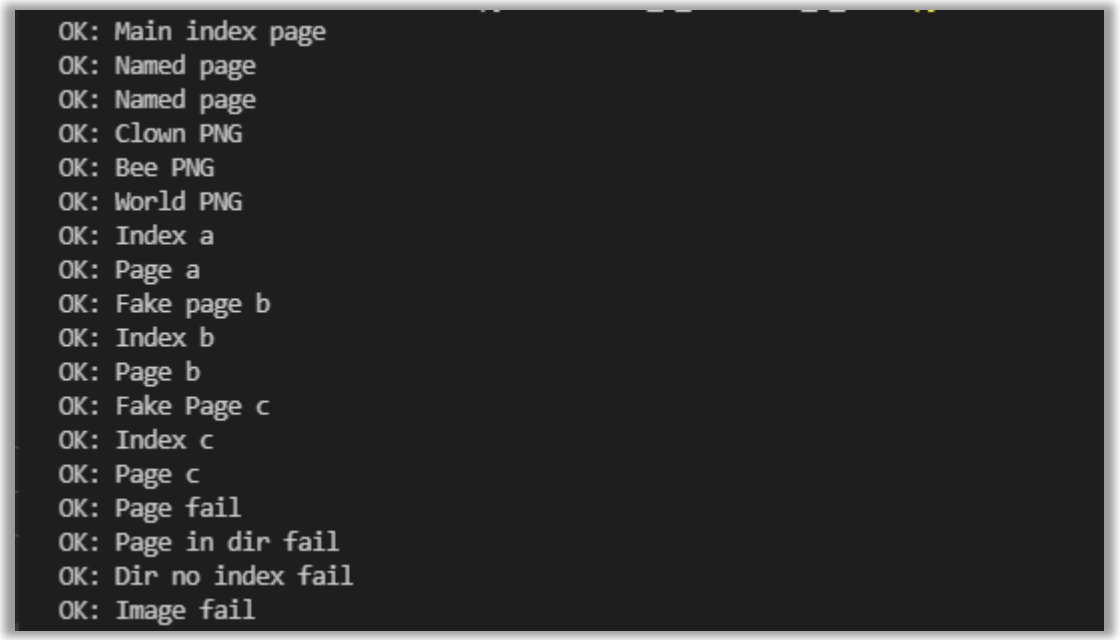

Figure 4. All tests passed

The process was somewhat the same for the bee as well however in order to run the tests, the user needs to enable the server first and then go to the test file and run the test file itself. Both the server and the test should indicate GET requests and the results aforementioned above.

1.2 Exceptions

There are different exceptions handled in our code such as: NumberFormatExcepetion, IOException, SocketException, and NoSuchFileException, etc.

- **NumberFormatException** → the reason to as why this is handled is because to warn the user that the argument for the port should be a valid integer. This was the exception thrown in the test case of inputting an invalid port number.
- **IOException** → the reason to as why it was implemented was due to a number of reasons regarding the sending of the requested file to the user, whenever handling requests and their properties, if occurs during the shutting down of the socket connection and so forth.
- **SocketException** → the reason to as why it was handled was to determine whether there is an error creating or accessing the socket and if so, interrupt the thread.
- **NoSuchFileException** → the reason to as why it was handled was to mitigate the status code 404 or “Not Found” whenever the user asks for a non-existing file.

Problem 2 – Implementations of different errors

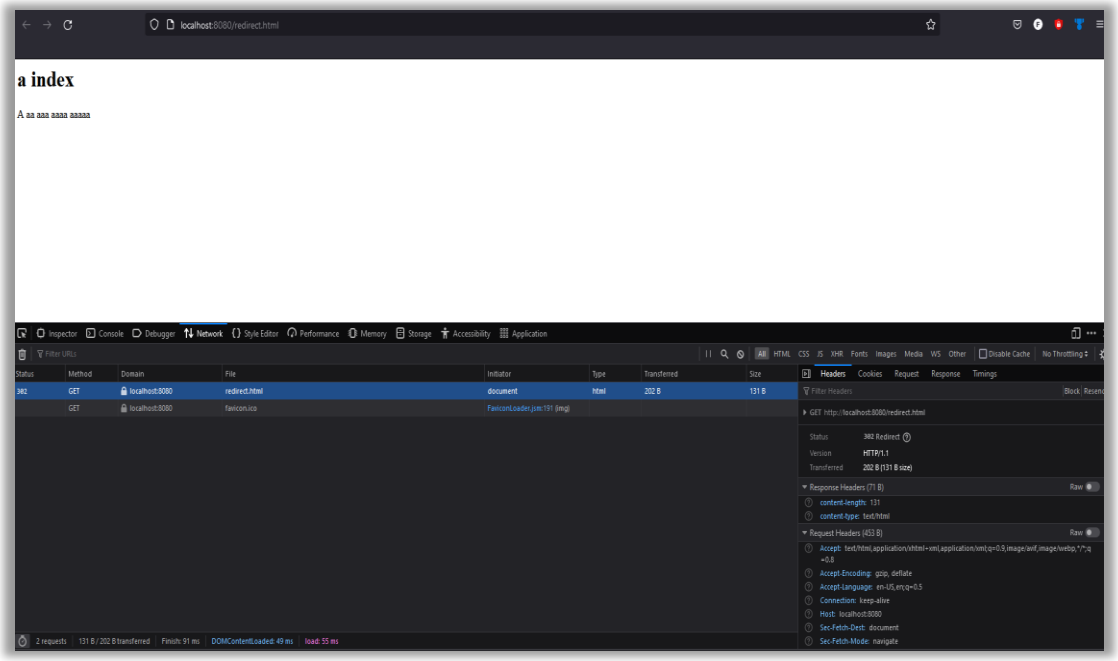

Figure 5. Redirecting working

In order to test whether redirecting works or not, as soon as “redirect.html” is requested it will redirect the user to “/public/a/index.html” instead and showing the “302 code” too. To test it, simply type in “localhost:[port chosen]/redirect.html/” and it shall provide the same results.

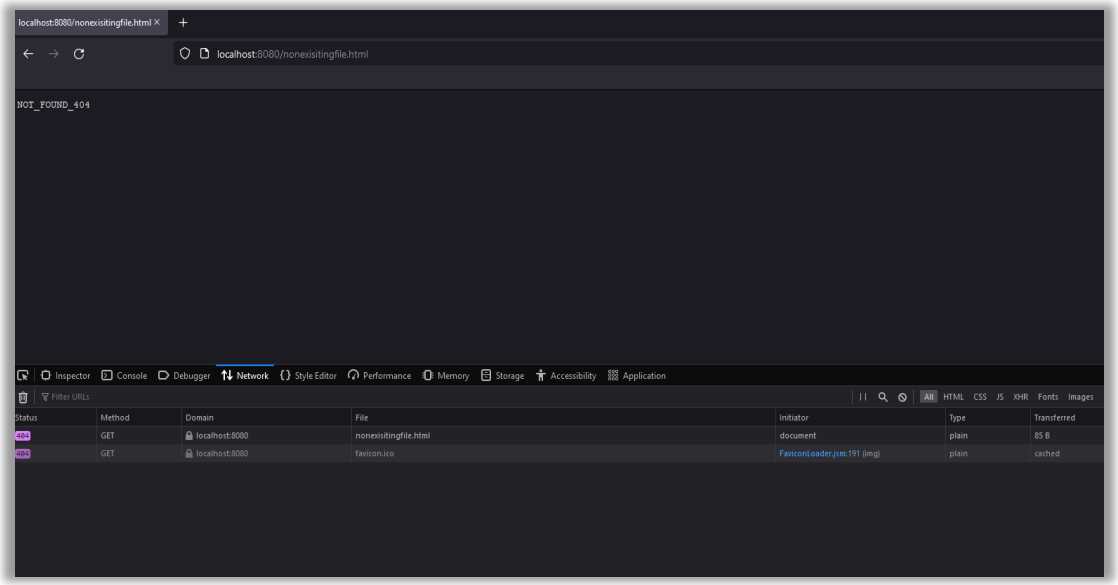

Figure 6. Not found error working.

In order to test this, a non-exisisting page was requested which lead to receiving the error in question. To test this, simply type in “localhost:[port chosen]/non-exisisting.html/” and it shall provide the same results as seen above.

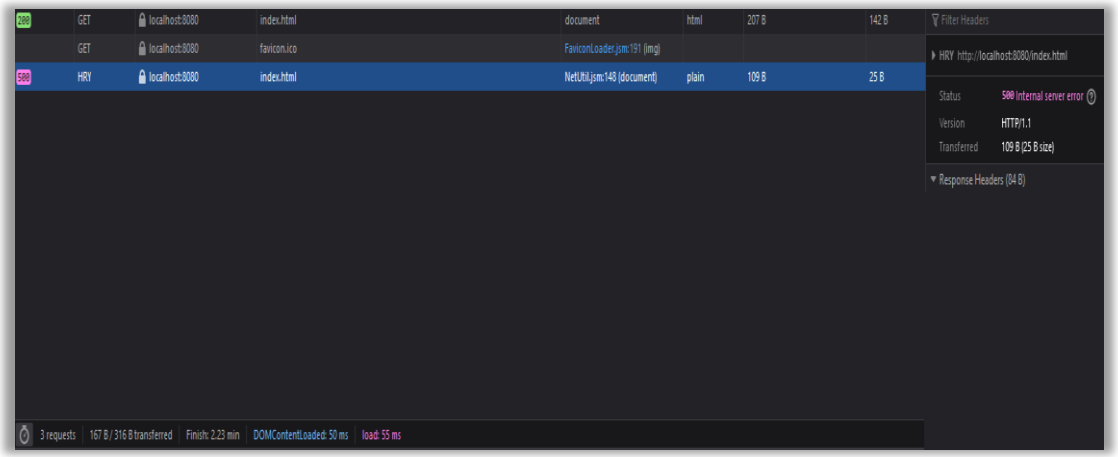

Figure 7. Internal server error working

In order to test this it is important to keep in mind that internal server usually occur whenever a server encounters a situation it does not know how to process therefore refactoring was done to a request with the help of inspection tools of the browser which lead to the error seen above. This is one of the ways this error can be tested.

VG Problem - Implement POST request

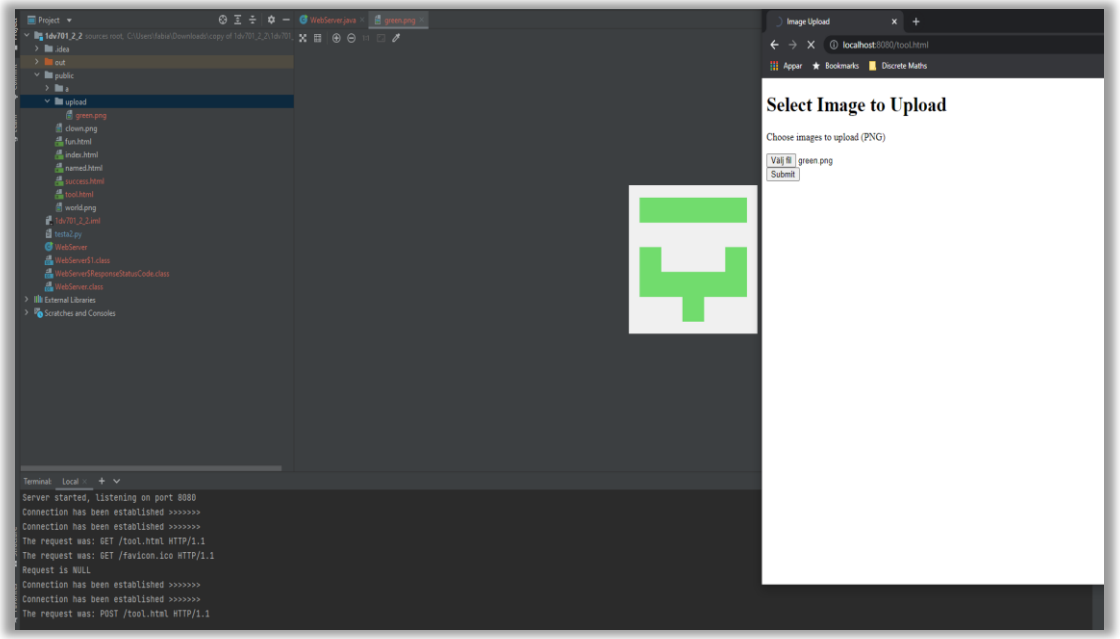

Figure 8. Server handling POST request for uploading .png images

The implementation of POST proved to be the most time consuming task of this assignment. In order to upload an image, the user needs to enter “localhost:[port chosen]/tool.html”, and a form page should be opened asking of the user to upload an image. The user should then wait approximately 5-10 seconds before click the “submit” button to submit an image since otherwise the upload might not happen (which we believe something related to multi-threads). In case it does not happen, the user then needs to either click on the submit button again or refresh the page and redo the process. If the process is done correctly, then the chosen image shall appear in the “/upload” folder as it can be seen in the figure above.

The potential reason for the demand of waiting is that we suspect due to the program was designed as multi-threaded and there must be existed something related to threadpool which we did not handled properly.We decided to keep the threadpool in our final hand-in version since we want to show that we have a good basic understanding of this technique which we acquired from Operating System (1DV512) course.

The challenge part of this implementation is related to properly handle “I/O Flow” to read and write the image in correct charset. The critical selection for “BufferedReader” and “BufferedWritter” is using “StandardCharsets.ISO_8859_1” which “UTF-8” could not transfer the image format correctly. For future research, we may try to implement the program with asynchronous call for separated thread and properly implementing the multi threads to interact with POST request handling.

Summary

Cooperation and time management was crucial during this assignment especially when encountering tasks that we do not have much experience with. The work was split between 50% for Yuyao Duan and 50% for Fabian Dacic. Both members were involved in all of the tasks due to their nature being similar.

To run the program:

- First compile the file by typing in "**javac WebServer.java**" in a terminal of your choice.

- Then run it by giving a desired port number between *0* and *65535* and "public" the root folder i.e "**java WebServer 8080 public**".

- An output in the terminal will be shown indicating either success or failure to launch the program.