Linnéuniversitetet
Kalmar Växjö

Report

Assignment 2

1DV701

Author: Djordje Dimitrov
Semester: Spring 2023
Email dd222im@student.lnu.se

Contents

1 Problem 1

1.1 Discussion

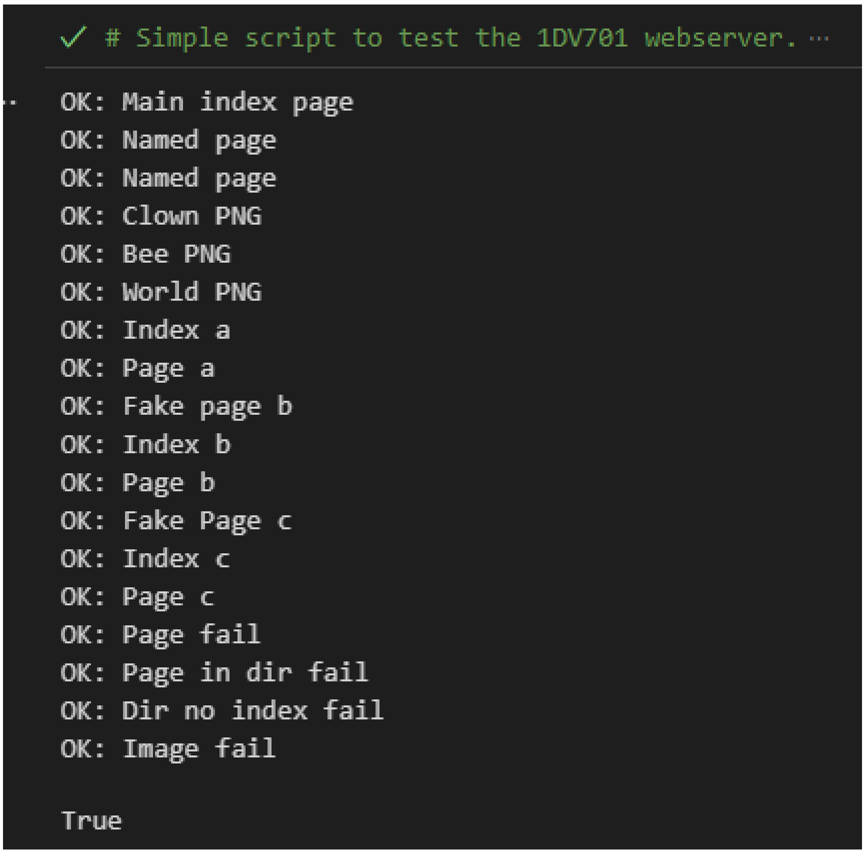
An overview of my solution:

- I call the argsCheck() method that checks the program arguments that are used as an input from the console. If the arguments are written in the correct form, the server starts and waits for the clients to connect. If not, the server prints the error messages and exits the application.
- After starting the server, the main method enters the while(true) loop that can accept multiple client connections. When a client opens the browser, the loop starts with

  ```
  try (Socket clientSocket = serverSocket.accept())
  ```

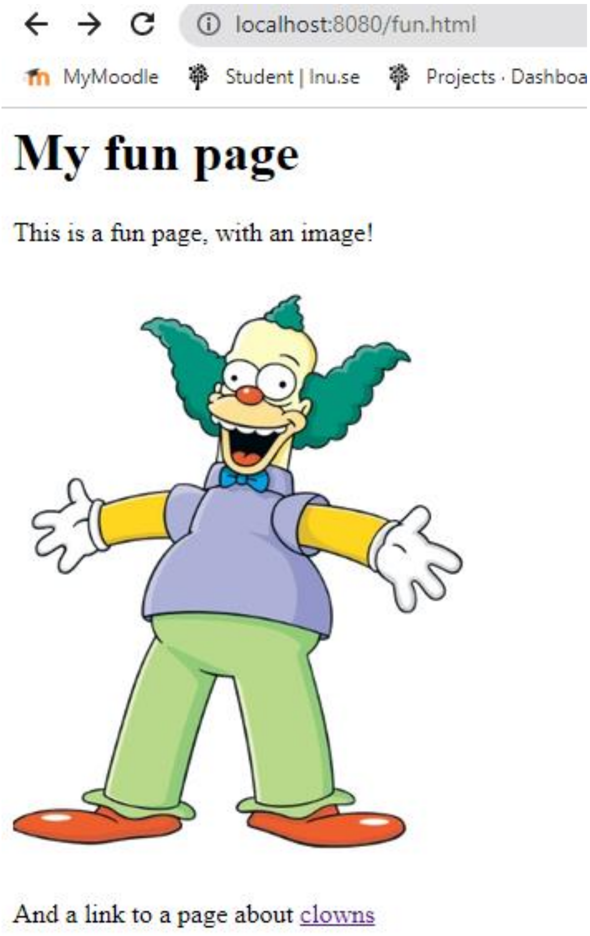
  and from there creates the Client object and passes the clientSocket parameter inside the object. The new thread is created for the connected client and from there the main loop waits for the thread to complete the work before finishing the loop.
- The Client class contains the entire logic for printing and showing the output inside the browser.It reads the input stream, separates the data in method, resource and version strings.
- Different outputs are being printed out, depending on the file extension and if the file exists or not.
- After the initial printing for the Requested file/item, the application checks the file path, and depending on the file, it writes the output and shows the data inside the browser window.
- setHeader() method I used in order to make the final printing in the console, I collect the data inside the method, such as response type, content length, content type, and current date and time. This method is used at the end of the code when making the last printouts.
- The client class handles the “IOException” in several places, for example when reading or writing to the socket output stream, or when attempting to read bytes from a file. The exceptions are handled by outputting the 404 error message, or 500 error message.
- The validatePath() method inside the Client class ensures that the user can’t escape the public directory.
- Javadoc comments are added on the methods and where necessary.

1.2 Test results



```
✓ # Simple script to test the 1DV701 webserver. ···

OK: Main index page
OK: Named page
OK: Named page
OK: Clown PNG
OK: Bee PNG
OK: World PNG
OK: Index a
OK: Page a
OK: Fake page b
OK: Index b
OK: Page b
OK: Fake Page c
OK: Index c
OK: Page c
OK: Page fail
OK: Page in dir fail
OK: Dir no index fail
OK: Image fail

True
```

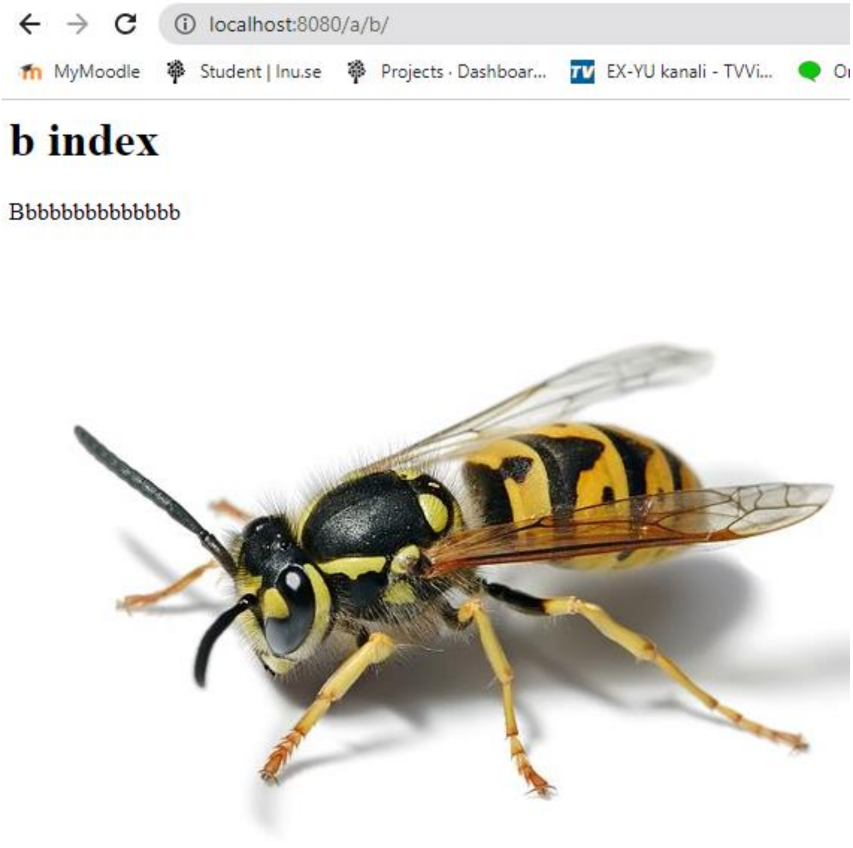
1.3 Screenshots



My fun page

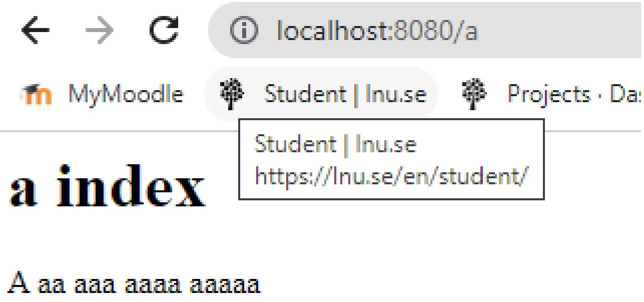
This is a fun page, with an image!

And a link to a page about clowns



b index

Bbbbbbbbbbbbbb



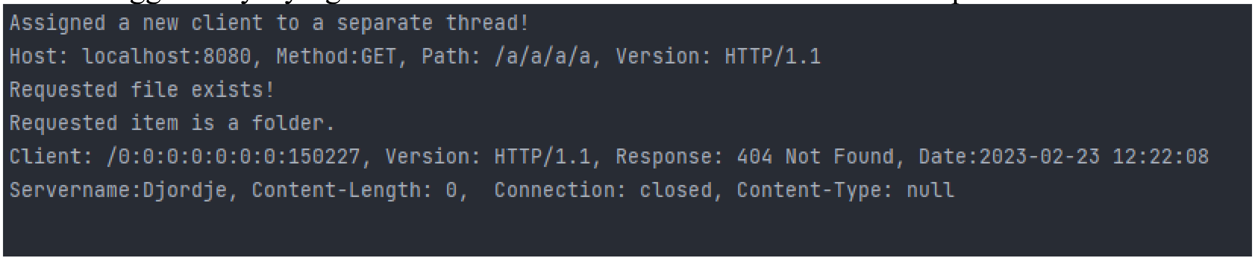
a index

A aa aaa aaaa aaaaa

2 Problem 2

2.1 Server not found error 404

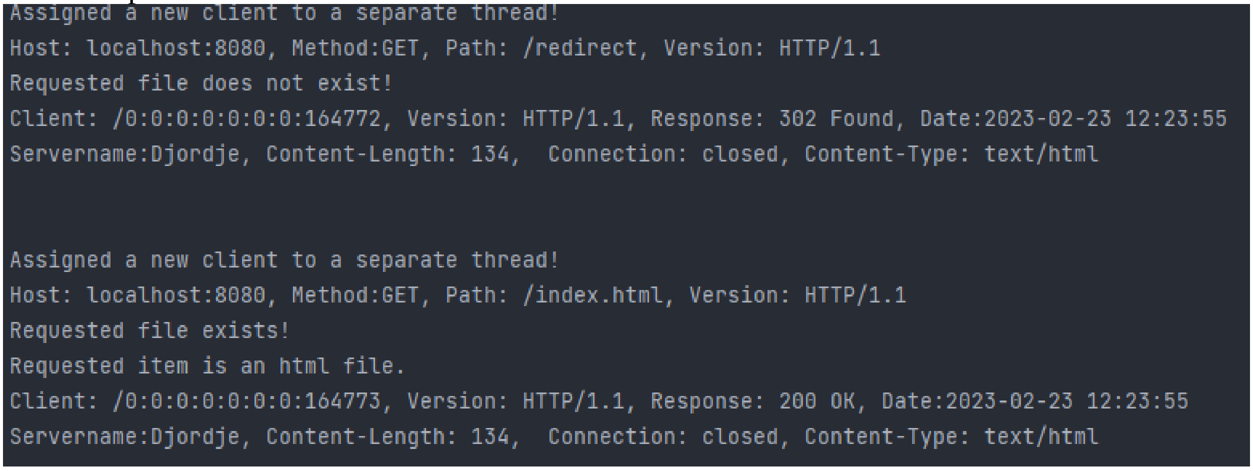
Can be triggered by trying to access the file that does not exist inside the ‘public’ folder.



```
Assigned a new client to a separate thread!
Host: localhost:8080, Method:GET, Path: /a/a/a/a, Version: HTTP/1.1
Requested file exists!
Requested item is a folder.
Client: /0:0:0:0:0:0:0:1:50227, Version: HTTP/1.1, Response: 404 Not Found, Date:2023-02-23 12:22:08
Servername:Djordje, Content-Length: 0,  Connection: closed, Content-Type: null
```

2.2 Redirected response 302

Can be triggered by typing localhost:port/redirect. Redirects to the homepage localhost:port/index.html.



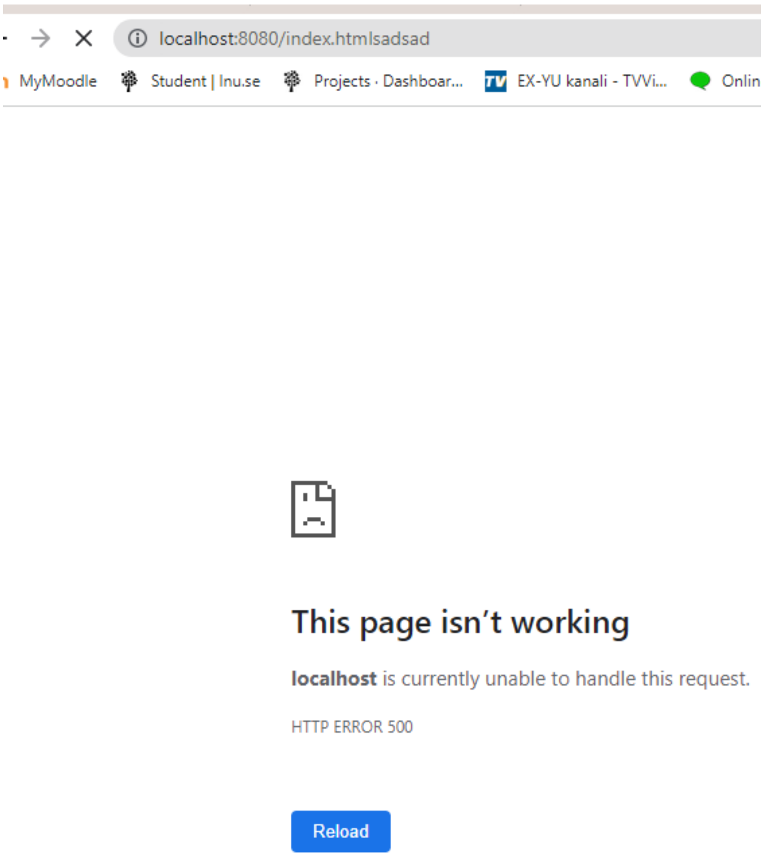
```
Assigned a new client to a separate thread!
Host: localhost:8080, Method:GET, Path: /redirect, Version: HTTP/1.1
Requested file does not exist!
Client: /0:0:0:0:0:0:0:1:64772, Version: HTTP/1.1, Response: 302 Found, Date:2023-02-23 12:23:55
Servername:Djordje, Content-Length: 134,  Connection: closed, Content-Type: text/html


Assigned a new client to a separate thread!
Host: localhost:8080, Method:GET, Path: /index.html, Version: HTTP/1.1
Requested file exists!
Requested item is an html file.
Client: /0:0:0:0:0:0:0:1:64773, Version: HTTP/1.1, Response: 200 OK, Date:2023-02-23 12:23:55
Servername:Djordje, Content-Length: 134,  Connection: closed, Content-Type: text/html
```

2.3 Internal server error 500

Can be triggered by calling the method on line 128 inside the Client class

```
byte[] data = Files.readAllBytes(Path.of(filePath.toUri()));
```

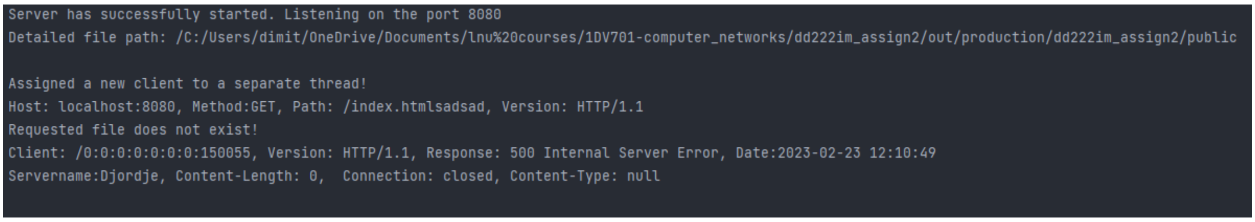
to a non-existing path such as http://localhost:8080/index.htmlsadsad
The method tries to read the bytes but fails since the file does not exist, and therefore triggers the internal server error on line 157 inside the Client class. I could trigger this error response only by making an error in the code deliberately.



This page isn't working

localhost is currently unable to handle this request.

HTTP ERROR 500



```
Server has successfully started. Listening on the port 8080
Detailed file path: /C:/Users/dimit/OneDrive/Documents/lnu%20courses/1DV701-computer_networks/dd222im_assign2/out/production/dd222im_assign2/public

Assigned a new client to a separate thread!
Host: localhost:8080, Method:GET, Path: /index.htmlsadsad, Version: HTTP/1.1
Requested file does not exist!
Client: /0:0:0:0:0:0:0:150055, Version: HTTP/1.1, Response: 500 Internal Server Error, Date:2023-02-23 12:10:49
Servername:Djordje, Content-Length: 0,  Connection: closed, Content-Type: null
```