



Linnéuniversitetet

Kalmar Vxjö

Report

Assignment 2

IDV701



Author: Etkä Kocak
Semester: Spring 2024

Contents

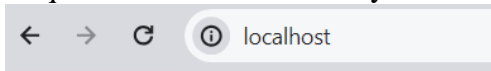
1 Problem 1 _____ **I**
 1.1 Discussion.....I

2 Problem 2 _____ **III**
 2.1 Discussion..... III

3 Problem 3 _____ **IIII**
 3.1 Discussion.....IIII

1 Problem 1

Request of the main directory:



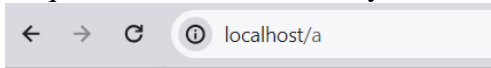
localhost

Server started on port 80
Server source file exists!
Detailed path: public
New client connected: 0:0:0:0:0:0:0:1:65384
Method: GET, Path: /, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 16:59:31 MEZ
Server: my_web_server
Content-Length: 134
Connection: close
Content-Type: text/html

Main index page

This is the main index page!

Request of a named directory:



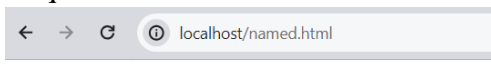
localhost/a

New client connected: 0:0:0:0:0:0:0:1:65410
Method: GET, Path: /a, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 17:05:28 MEZ
Server: my_web_server
Content-Length: 123
Connection: close
Content-Type: text/html

a index

A aa aaa aaaa aaaaa

Request of a named HTML file:



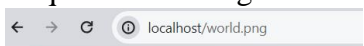
localhost/named.html

New client connected: 0:0:0:0:0:0:0:1:65420
Method: GET, Path: /named.html, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 17:07:31 MEZ
Server: my_web_server
Content-Length: 153
Connection: close
Content-Type: text/html


What's my name, name, name

I tried to join a ping-pong club

Request of an image:



localhost/world.png



New client connected: 0:0:0:0:0:0:0:1:65426
Method: GET, Path: /world.png, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 17:09:34 MEZ
Server: my_web_server
Content-Length: 14787751
Connection: close
Content-Type: image/png

1.1 Discussion

How does the WebServer.java I developed work:

main: Starts the server and listens for client requests in an infinite loop. Then calls the **handleRequest** method for each connection by also sending the request.

handleRequest: Reads the HTTP request from the client, reads the file from disk accordingly, and creates the HTTP response. If the requested file does not exist or another error occurs, it sends a response with the appropriate error message. It also prints the generated HTTP response with detailed information to the console.

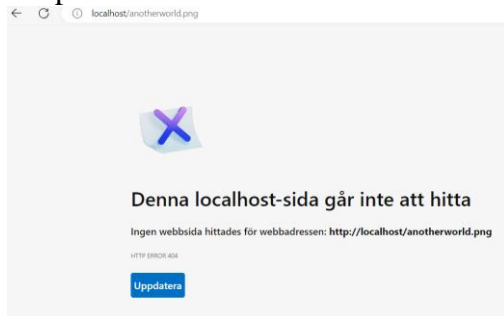
sendResponse: Sends a simple HTTP response to the client for exception handling, used in case of error.

Safety against escaping the root:

1. Path Normalization: The file path requested by the user is normalized. In this way, it prevents users from leaving and going outside the directory.
2. Starting Path Check: The requested file directory is checked again and if the file's path does not start with the specified "public" directory or the file does not exist, access is blocked.

2 Problem 2

Response 404:



```
New client connected: 0:0:0:0:0:0:0:1:49576
Method: GET, Path: /anotherworld.png, Version: HTTP/1.1
Requested file does not exist within the public directory!
Response Headers:
HTTP/1.1 404 Not Found
```

Response 302:

```
New client connected: 0:0:0:0:0:0:0:1:50483
Method: GET, Path: /redirect.html, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 302 Found
Location: https://www.google.se/
Date: Sa., 24 Feb. 2024 20:16:03 MEZ
Server: my_web_server
Content-Length: 310
Connection: close
Content-Type: text/html
```

2.1 Discussion

The 404 response code means that the requested resource could not be found on the server. On my Java web server, a 404 code is returned if the file or directory the user requested does not exist or if the security rule is violated.

Testing:

Open a web browser and request a file or directory that does not exist. For example: "localhost/anotherworld.png". Observe receiving a 404 response from the server.

The 302 response code means that the requested resource has been temporarily moved to another URL. For example, I redirected requests to /redirect to google.se.

Testing:

Open a web browser and request: "localhost/redirect.html". Observe that your browser redirects you to the new specified URL. This means you have been successfully redirected with a 302 response code.

3 Problem 3

When login successful:

```
New client connected: 0:0:0:0:0:0:1:51125
Method: GET, Path: /login.html, Version: HTTP/1.1
Requested file exists!
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 21:37:02 MEZ
Server: my_web_server
Content-Length: 656
Connection: close
Content-Type: text/html

New client connected: 0:0:0:0:0:0:1:51126
Method: POST, Path: /fun.html, Version: HTTP/1.1
Requested file exists!
Username from HTML:etka
Password from HTML:etka123
Validate:true
Response Headers:
HTTP/1.1 200 OK
Date: Sa., 24 Feb. 2024 21:37:18 MEZ
Server: my_web_server
Content-Length: 232
Connection: close
Content-Type: text/html
```

When login not successful:

```
New client connected: 0:0:0:0:0:0:1:51151
Method: POST, Path: /fun.html, Version: HTTP/1.1
Requested file exists!
Username from HTML:rndm
Password from HTML:dfdfs
Validate:false
Response Header:HTTP/1.1 401 Unauthorized
...
```

3.1 Discussion

As can be seen from the images, I created an HTML file containing a login page. If the correct username and password are entered here, the user will be directed to the fun.html file. (e.g. username: etka password: etka123). However, if the wrong password is typed, the user will receive a 401 response code and will not be redirected to the page.

I set up the login system in WebServer as follows:

If a "POST" request comes to the "fun.html" file from the created login file, the parsePostData method parses and obtains the "username" and "password" data from the login page. This data is then sent to the validateUser method, which compares this data with the data in the "users.txt" file where the predefined user information is stored. If the username and password match, a "200 OK" response is sent to the client and the redirect is successful, otherwise a "401 Unauthorized" response is returned.

This system provides a simple user authentication mechanism. Users can log in to the system with their predefined username and password. If the login is successful, the requested content is presented; If it fails, access is denied.