Web Server

CSC667-java-brewers☕

Alexander Chau (921121393) & Michael Satumba (916849250)

CSC 0667-02 Internet Application Design and Development Fall 2021

San Francisco State University

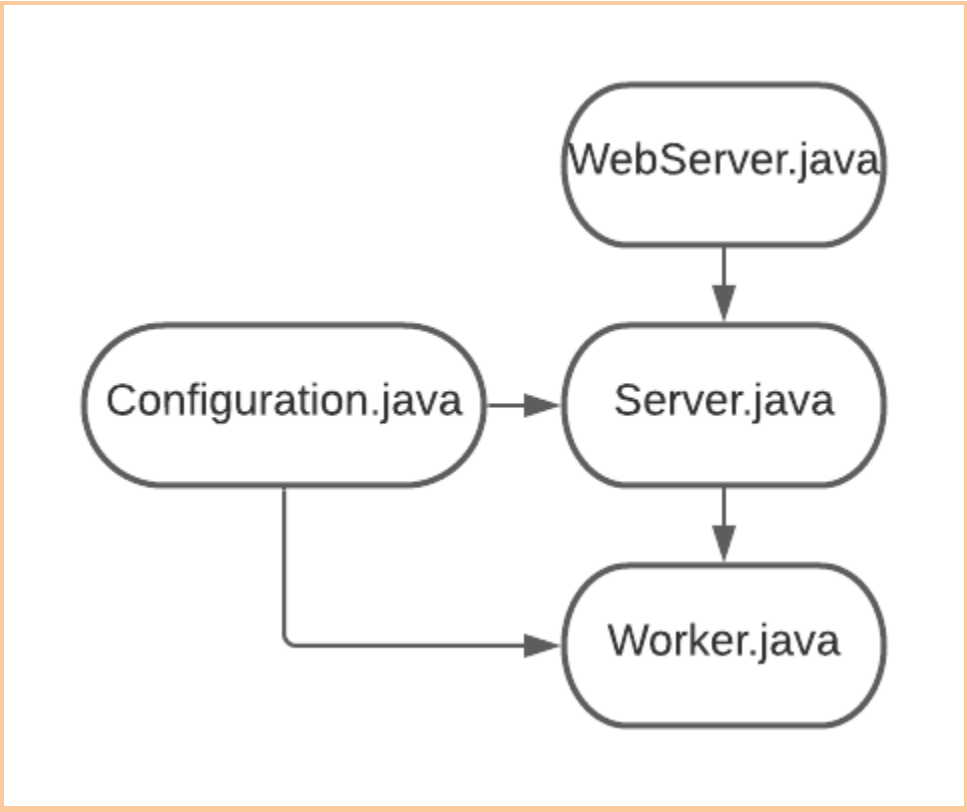
San Francisco, CA

San Francisco State University

Github Repo Link

https://github.com/sfsu-csc-667-fall-2021/web-server-alex-michael-repo

Web Server Architecture



Description:

WebServer.java serves as the driver code for the server application. This creates a new Server instance. From Server, the port, which is parsed using a Configuration.java object and stored in a HttpConfig.java object, is opened and listening. When a connection request is received, a thread Worker.java is spawned and managed by the threadpool, and the socket is passed to this thread. It is in workers where most of the business logic is handled. A worker

instantiates Configuration.java again to parse the configuration files (both httpd.conf and mime.types) and the remaining .java files (HtAccess, Htpassword, RunScript) in order to handle authentication, authorization, and script execution.

web-server/ Directory: (top level)

WebServer.java - Driver code

server/ directory:

Server.java - Open port, listen, and create new threads

Worker.java - Handle resource building, access check, and sending requests and responses

server/config/ directory:

Configuration.java - Read in config files and store in data structures

HttpdConfig.java - Store httpd.config directives and provide API retrieval for this data

MimeTypes.java - Store mime.types extensions and provide API retrieval for this data

HtAccess.java - Store .htaccess fields and provide API retrieval for this data

Htpassword.java - Parses and stores .htpasswd fields, provides API retrieval for this data, and handles encryption/decryption

public_html directory:

RunScript.java - Executes and perl or shell script

**Note: RunScript.java was shown to be in the public_html/cgi-bin/ directory, but this caused problems with the package declaration. Online troubleshooting

revealed no solution, so I moved this into the root of the public_html directory. It may be more reasonable to keep it in server/ instead.**

Problems Encountered

Alex:

- Problem: httpd.config contained inconsistent lines with varying content (e.g. Alias /ab/ <path> vs. Listen <port>)
    - Solution: Use a Concurrent HashMap with conditionals and string splitting to create the data structure needed
- Problem: Similar to httpd.conf, mime.types varied in file-extension count. There was also the issue of many-to-one mappings.
    - Solution: Used a HashMap<String[], String> and iterated through the key array
- Problem: NullPointerException when multithreading
    - Solution: Started using ExecutorService
- Problem: PrintWriter was not working for images.
    - Solution: Switched to OutputStream
- Problem: In general, learning and using the stream objects was tedious and time-consuming.
    - Solution: Deal with it and read documentation.
- Problem: NullPointerException due to an empty request being accepted by the thread. The web server does not crash due to this.

- Still ongoing, currently is being caught and the stack trace printed to stdout.

Michael: I had trouble implementing the response from the server. I also wasn't familiar with Java, so I had to learn the basics to be useful for the project. I built a super simple web server to just get familiar with Java. I utilized Google, StackOverFlow, and YouTube. Moreover, I had issues implementing simple caching and logging. I utilized online sources to find solutions.

Difficulty Discussion

Alex: I think the most difficult aspect of the project was debugging. I had trouble with my debugger in both VSCode and IntelliJ Idea, so I was limited to manual debugging with “cout” statements. I didn’t want to spend more time than I already had on trying to get the debugger working, so this was a tradeoff I had to make. Thus, it was difficult narrowing down and fixing exceptions in general. I think handling script execution and authentication were the most difficult, or required more time than I had to implement.

Michael: It was difficult to get a response from the server because I didn't know how to solve the error Postman kept giving me: `Error: Parse Error: Invalid header`

token. Another problem was that the proxy in the Postman App was not configured to test localhost:8080. Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.isBlank()" because "<local5>" is null at WebServer.listenForReq(WebServer.java:33) at WebServer.main(WebServer.java:12). So I had to check "add a custom proxy configuration" and then I had to put : Proxy Server 127.0.01 : 8080. Moreover, Postman was giving persistent issues, I utilized curl for the majority of the project.

Another issue that was difficult was implementing the proper request with their responses. For example, for the POST and PUT methods they had fifth and sixth lines in their request that displayed the content-length and content-type. But for the other methods those lines were not included. I solved it by using a switch statement and declaring the fifth and sixth lines within the case.

One more thing was testing GET via the browser and curl. In curl everything worked as I expected. But in the browser, the request would hang indefinitely until the server was closed and then it would finally show the response.

Test Plan

Curl:

GET - curl http://localhost:8080/

Expected response:

Index.html file

HEAD - curl -I http://localhost:8080/

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

POST - curl -d "user=user1&pass=abcd" -X POST http://localhost:8080/

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:31:19.451021
Server: Chau & Satumba
Content-Length: 20
Content-Type: application/x-www-form-urlencoded

PUT - curl -d "user=user1&pass=abcd" -X PUT http://localhost:8080/

Expected response:

HTTP/1.1 201 Created
Date: 2021-09-27T22:30:39.584744
Server: Chau & Satumba
Content-Length: 20
Content-Type: application/x-www-form-urlencoded

DELETE - curl -d "user=user1&pass=abcd" -X DELETE http://localhost:8080/

Expected response:

HTTP/1.1 204 No Content
Date: 2021-09-27T22:29:32.287815
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

Telnet:

GET / HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

GET /images/sushi.jpg HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

GET /ab/ HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

GET /~traciely/ HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

GET /cgi-bin/ HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>

Content-Type: text/html; charset=utf-8

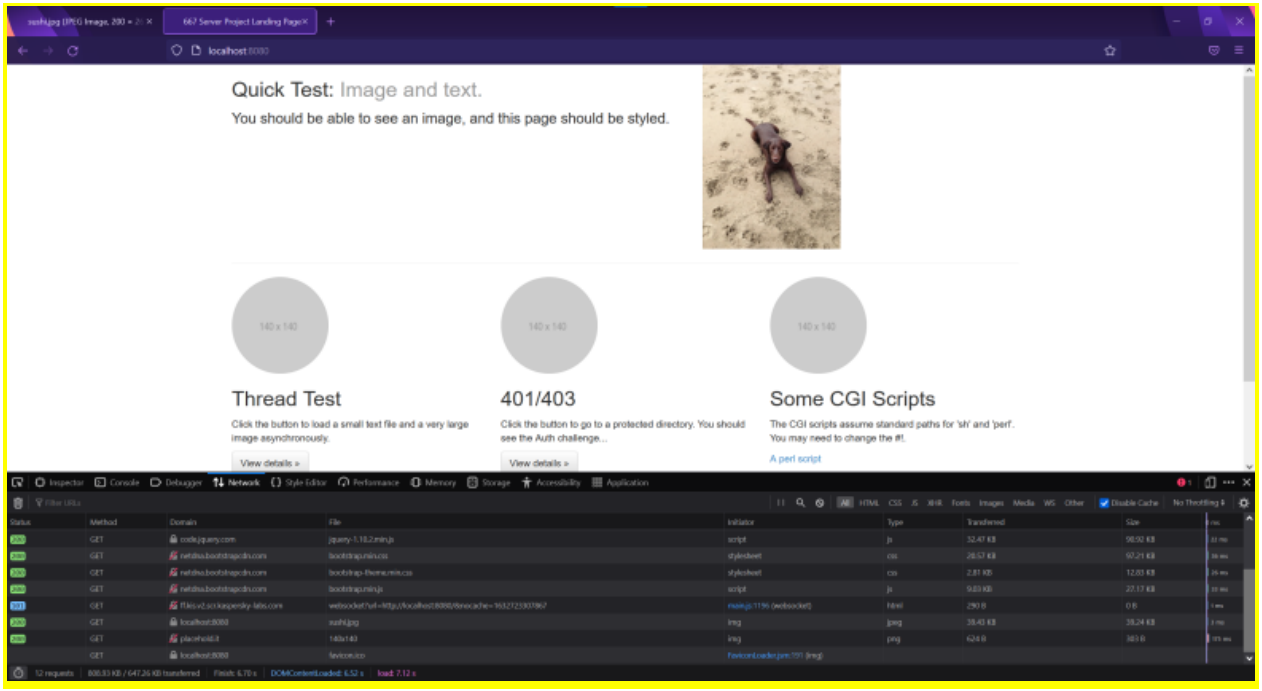
GET /cgi-bin/perl_env HTTP/1.1 Host: localhost:8080

Expected response:

HTTP/1.1 200 OK
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

Browser Tests:

http://localhost:8080/



http://localhost:8080/g

Expected response:

HTTP/1.1 404 Not Found
Date: 2021-09-27T22:28:34.591909
Server: Chau & Satumba
Content-Length: <content-length>
Content-Type: text/html; charset=utf-8

Grading Rubric

Category	Description	Out of __ points	__ points given
Code Quality	Code is clean, well formatted (appropriate white space and indentation)	5	3
	Classes, methods, and variables are meaningfully named (no comments exist to explain functionality - the identifiers serve that purpose)	5	4
	Methods are small and serve a single purpose	3	3
	Code is well organized into a meaningful file structure	2	2

Total: 12 /15

Category	Description	Out of __ points	__ points given
Documentation	A PDF is submitted that contains:	3	3
	Full names of team members	3	3
	A link to github repository	3	3
	A copy of this rubric with each item checked off that was completed (feel free to provide a suggested total you deserve based on completion)	1	1
	Brief description of architecture (pictures are handy here, but do not re-submit the pictures I provided)	5	5
	Problems you encountered during implementation, and how you solved them	5	5
	A discussion of what was difficult, and why	5	5
	A thorough description of	5	5

	your test plan (if you can't prove that it works, you shouldn't get 100%)		

Total: 30/30

Category	Description	Out of __ points	__ points given
Functionality - Server	Starts up and listens on correct port	2	2
	Logs in the common log format to stdout and log file	2	0
	Multithreading	5	5

Total: 7/10

Category	Description	Out of __ points	__ points given
Functionality - Responses	200	2	2
	201	2	2
	204	2	2
	400	2	2
	401	2	0
	403	2	0
	404	2	2
	500	2	2
	Required headers present (Server, Date)	1	1
	Response specific headers present as needed (ContentLength, Content-Type)	2	2
	Simple caching (HEAD with If-Modified-Since results in 304 with Last-Modified header, Last-Modified header sent)	1	0
	Response body correctly sent	3	3

Total: 18/23

Category	Description	Out of __ points	__ points given
Functionality - Mime Types	Appropriate mime type returned based on file extension (defaults to text/text if not found in mime.types)	2	2

Total: 2/2

Category	Description	Out of __ points	__ points given
Functionality - Config	Correct index file used (defaults to index.html)	1	1
	Correct htaccess file used	1	1
	Correct document root used	1	1
	Aliases working (will be mutually exclusive)	3	3
	Script Aliases working (will be mutually exclusive)	3	3
	Correct port used (defaults to 8080)	1	1
	Correct log file used	1	1

Total: 11/11

Category	Description	Out of __ points	__ points given
CGI	Correctly executes and responds	4	4
	Receives correct environment variables	3	3
	Connects request body to standard input of cgi process	2	2

Total: 9/9

Grand Total: 89/100