COMP 2322 Project Report

Multi-thread Web Server

Student Name: Guanlin Jiang
Student ID: 21093962D

1. Introduction

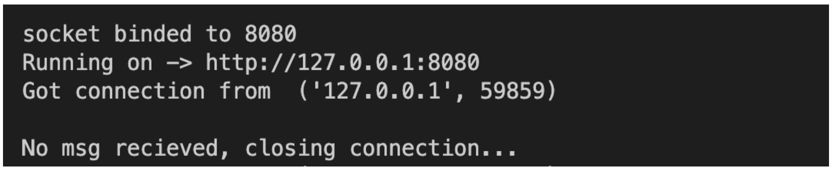
The Multi-thread Web Server is designed to send the HTML file to the web and open it with the port, also can send multiple times request – put and get to process it. The website status codes and the header in web server system will be sent to the target clients. For this web server, user only need to put the file to the folder, which is more convenience for user. Also, when website is not working, the web status code will be changed and display to user.

2. Web Server Functions

2.1 Ports

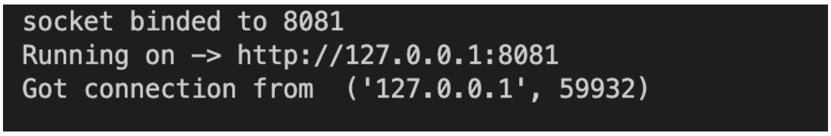
In this web server, when the user launches the web server, the server will be judging the current port is used or not, if not used, web server will be posting the website to that port. In default, the address will be 127.0.0.1 and the port will be 80. If the port used, the program will be changed to another port and remind user the port is changed and launch in another one, but the address still will be the original.

For example:



```
socket binded to 8080
Running on -> http://127.0.0.1:8080
Got connection from  ('127.0.0.1', 59859)

No msg recieved, closing connection...
```

[Website run on Port 8080]



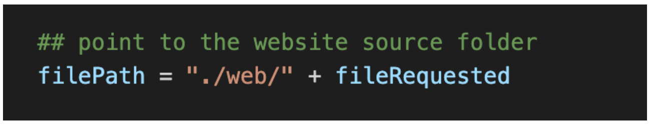
```
socket binded to 8081
Running on -> http://127.0.0.1:8081
Got connection from  ('127.0.0.1', 59932)
```

[Website run on Port 8081, if not close before one]

2.2 Hosting Files

The location of file host place will be under the web server python file which have a folder named “web”. User can put the source files of the website to that folder, if the source file has index.html file, the web server will be point to that file when the server initiate.

For example:



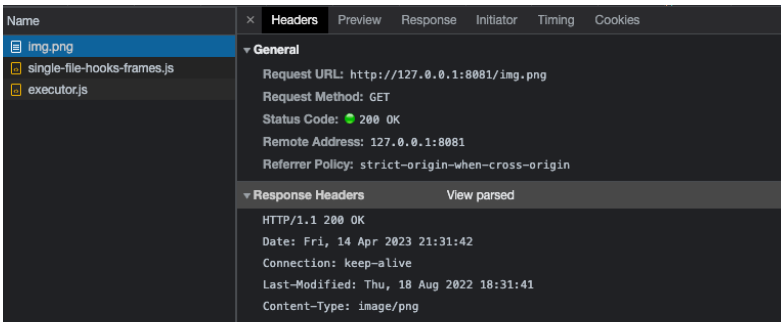
```
## point to the website source folder
filePath = "./web/" + fileRequested
```

[The folder of the website source code]

2.3 Website Header

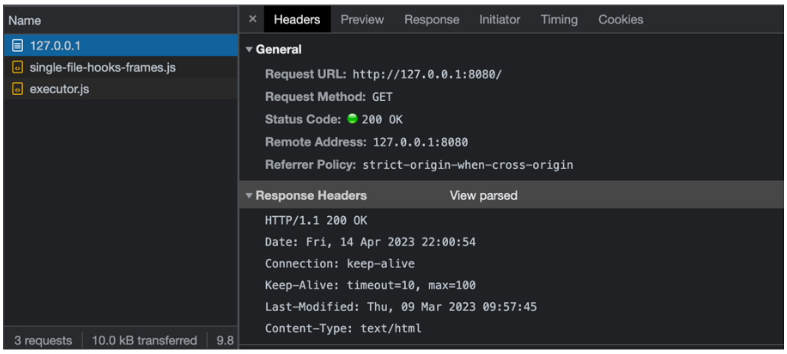
In this web server, when user visit the website, the web server will be sent the website header to client, which include the time user visit, the status code of the website to identify the website is successful get or the request is unsuccessful, the request file of HTTP version which will be display in header and the connection type which can know this request is for text (html) or the image (png or jpg file). Also, the connection status is important to add in, that can judge this request is keep-alive or not, and the modified time with date will be add to header, when the user request the file, after the If-Modified-Since header will be send to web server, if the timestamp is not equal to the file modified date, the status code will be 304.

For example:



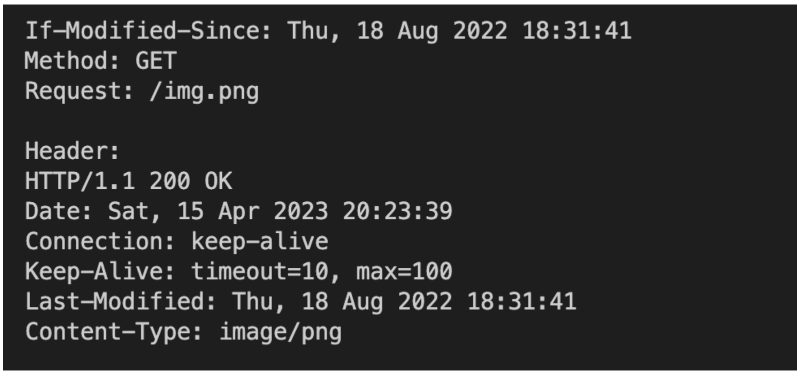
```
Request URL: http://127.0.0.1:8081/img.png
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:8081
Referrer Policy: strict-origin-when-cross-origin

HTTP/1.1 200 OK
Date: Fri, 14 Apr 2023 21:31:42
Connection: keep-alive
Last-Modified: Thu, 18 Aug 2022 18:31:41
Content-Type: image/png
```

[The header of Image file]



```
Request URL: http://127.0.0.1:8080/
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:8080
Referrer Policy: strict-origin-when-cross-origin

HTTP/1.1 200 OK
Date: Fri, 14 Apr 2023 22:00:54
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: Thu, 09 Mar 2023 09:57:45
Content-Type: text/html
```

[The header of HTML file]



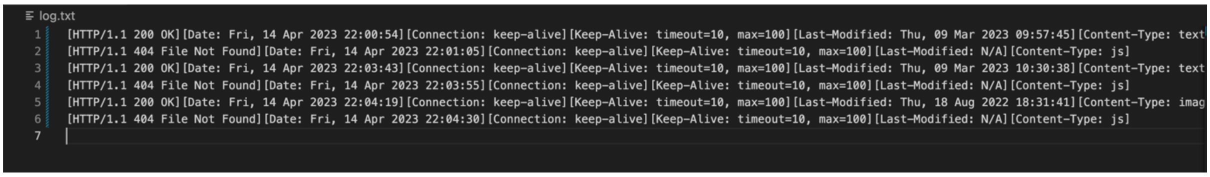
```
If-Modified-Since: Thu, 18 Aug 2022 18:31:41
Method: GET
Request: /img.png

Header:
HTTP/1.1 200 OK
Date: Sat, 15 Apr 2023 20:23:39
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: Thu, 18 Aug 2022 18:31:41
Content-Type: image/png
```

[The If-Modified-Since & Last-Modified]

2.4 Log File

In this web server, the header for each request will be send to the client, also will be store in the log file, the log file which will be include the information – the state code, the time stamp, the connection is keep alive or not, and the connection type is belong to html text or get image of this request from client.

For example:



```
[HTTP/1.1 200 OK][Date: Fri, 14 Apr 2023 22:00:54][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 09 Mar 2023 09:57:45][Content-Type: text
[HTTP/1.1 404 File Not Found][Date: Fri, 14 Apr 2023 22:01:05][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
[HTTP/1.1 200 OK][Date: Fri, 14 Apr 2023 22:03:43][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 09 Mar 2023 10:30:38][Content-Type: text
[HTTP/1.1 404 File Not Found][Date: Fri, 14 Apr 2023 22:03:55][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
[HTTP/1.1 200 OK][Date: Fri, 14 Apr 2023 22:04:19][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 18 Aug 2022 18:31:41][Content-Type: imag
[HTTP/1.1 404 File Not Found][Date: Fri, 14 Apr 2023 22:04:30][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
```

[The log file writes by web server]

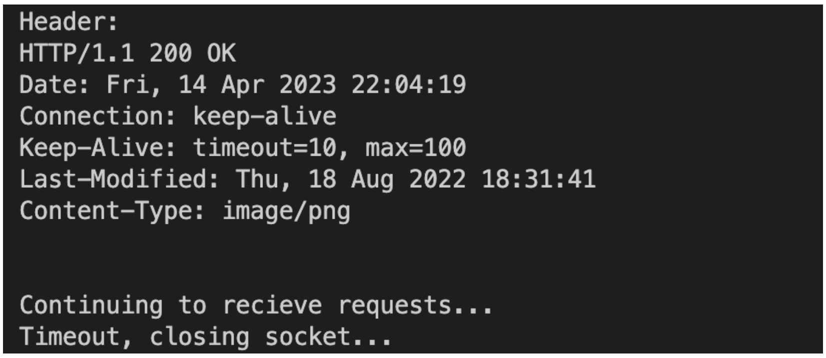
2.5 Multiple Thread

This function is the most important part of this web server, the client maybe will send many connection requests for get information or post information, and web server need to be handled those multiple requests simultaneously. In this function, I used the library - threading to make the multiple thread server connection and requestion.

2.6 Keep Alive

In the web server, I set the keep alive time of timeout to 10 seconds, and maximum for 100 seconds. When the content sent to client, the web server will be continuing to get the requests from client and also to judge the server and client connection is persistent connection or not, if no requests after 10 seconds, the connection will be timeout.

For example:



```
Header:
HTTP/1.1 200 OK
Date: Fri, 14 Apr 2023 22:04:19
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: Thu, 18 Aug 2022 18:31:41
Content-Type: image/png


Continuing to recieve requests...
Timeout, closing socket...
```

3. Web Server Structures

3.1 Data Structure

3.1.1 Array

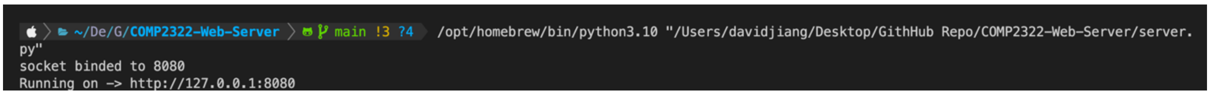
In this web server, I use the array to store the header information that will be store into the log file of the web server.

3.2 Program Structure

When user host the file and initial the web server, the server first will be checking the port is used or not, if used, the file will be to point to another socket port to routing. After, the server will be checking the website source file in the folder, if have index.html file, will be sent it to the web which use file open and read operation to do it. At the same time, the header will be created and when user visit the website, the header will be sent to the client.

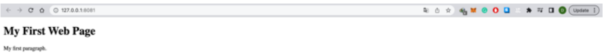
Here is Demo of the Web Server:

1. Run the server.py and release the website



```
 ~/De/G/COMP2322-Web-Server  main !3 ?4  /opt/homebrew/bin/python3.10 "/Users/davidjiang/Desktop/GithHub Repo/COMP2322-Web-Server/server.
py"
socket binded to 8080
Running on -> http://127.0.0.1:8080
```

2. Visit the website on web browser

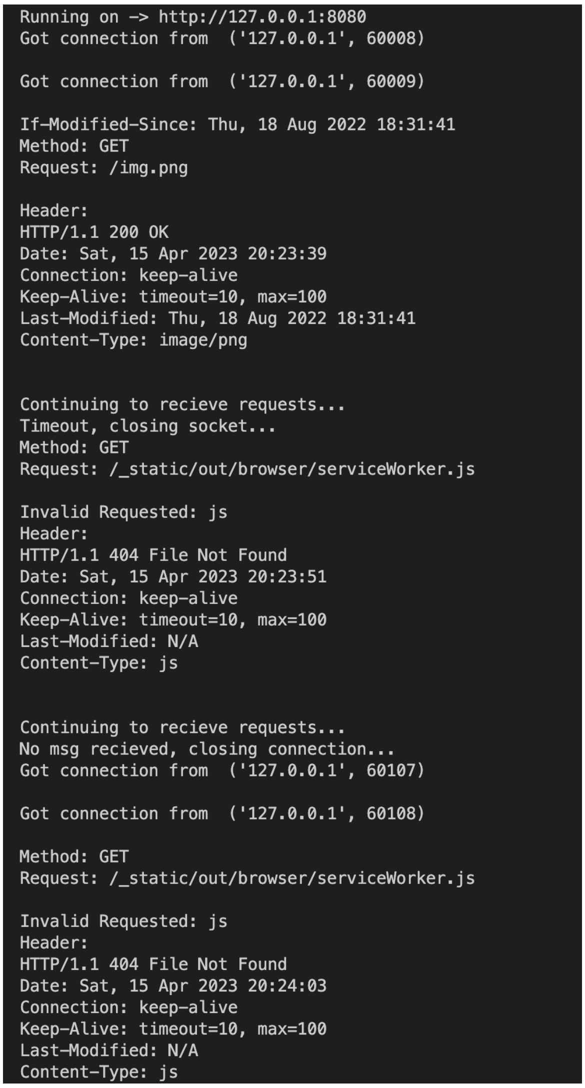


127.0.0.1:8081

My First Web Page

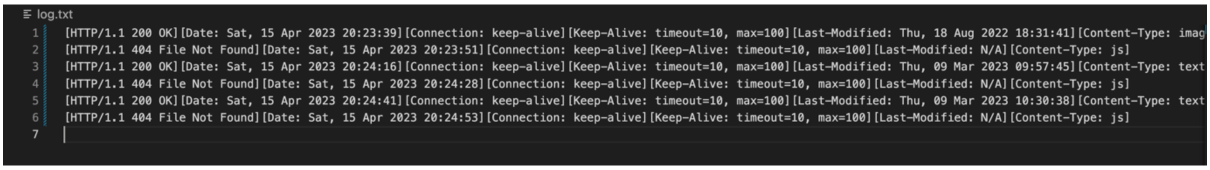
My first paragraph.

3. The output of the web server after the client connection



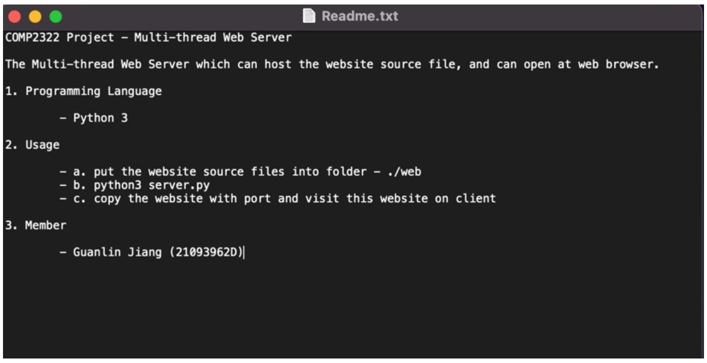
```
Running on -> http://127.0.0.1:8080
Got connection from  ('127.0.0.1', 60008)

Got connection from  ('127.0.0.1', 60009)

If-Modified-Since: Thu, 18 Aug 2022 18:31:41
Method: GET
Request: /img.png

Header:
HTTP/1.1 200 OK
Date: Sat, 15 Apr 2023 20:23:39
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: Thu, 18 Aug 2022 18:31:41
Content-Type: image/png


Continuing to recieve requests...
Timeout, closing socket...
Method: GET
Request: /_static/out/browser/serviceWorker.js

Invalid Requested: js
Header:
HTTP/1.1 404 File Not Found
Date: Sat, 15 Apr 2023 20:23:51
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: N/A
Content-Type: js


Continuing to recieve requests...
No msg recieved, closing connection...
Got connection from  ('127.0.0.1', 60107)

Got connection from  ('127.0.0.1', 60108)

Method: GET
Request: /_static/out/browser/serviceWorker.js

Invalid Requested: js
Header:
HTTP/1.1 404 File Not Found
Date: Sat, 15 Apr 2023 20:24:03
Connection: keep-alive
Keep-Alive: timeout=10, max=100
Last-Modified: N/A
Content-Type: js
```

Here is Web Server Log File:



```
[HTTP/1.1 200 OK][Date: Sat, 15 Apr 2023 20:23:39][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 18 Aug 2022 18:31:41][Content-Type: imag
[HTTP/1.1 404 File Not Found][Date: Sat, 15 Apr 2023 20:23:51][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
[HTTP/1.1 200 OK][Date: Sat, 15 Apr 2023 20:24:16][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 09 Mar 2023 09:57:45][Content-Type: text
[HTTP/1.1 404 File Not Found][Date: Sat, 15 Apr 2023 20:24:28][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
[HTTP/1.1 200 OK][Date: Sat, 15 Apr 2023 20:24:41][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: Thu, 09 Mar 2023 10:30:38][Content-Type: text
[HTTP/1.1 404 File Not Found][Date: Sat, 15 Apr 2023 20:24:53][Connection: keep-alive][Keep-Alive: timeout=10, max=100][Last-Modified: N/A][Content-Type: js]
```

Here is Web Server Readme:



```
COMP2322 Project - Multi-thread Web Server

The Multi-thread Web Server which can host the website source file, and can open at web browser.

1. Programming Language

        - Python 3

2. Usage

        - a. put the website source files into folder - ./web
        - b. python3 server.py
        - c. copy the website with port and visit this website on client

3. Member

        - Guanlin Jiang (21093962D)
```