Global SW

Cloud Computing

Dolgormaa

32215125

Github repository: <https://github.com/Dogii0/CloudComputing>

# HW1: Make your own web server

The objective of this homework was to create a simple web server using the socket API. The program is designed to listen on port 80, accept incoming connections, read the data from the port, parse the input and print out the data according to the HTTP GET request, prepare a response according to the GET request and send it to the client. This program was made using python and we would be checking the result on “localhost”.

## Implementation Details

The implementation is done in Python, utilizing the socket library and used threading for Multi-threading Support. (<https://docs.python.org/3/howto/sockets.html> )

ChatGPT usage: Multi-threading Support

### **Code Overview**

The provided Python code consists of two main functions: handle\_client() and main().

handle\_client(): This function is responsible for handling individual client connections. It reads the incoming data, parses the HTTP request, retrieves the requested file, prepares the HTTP response, and sends it back to the client.

main(): This function sets up the server socket, binds it to port 80, and listens for incoming connections. Upon accepting a connection, it creates a new thread to handle the client.

Below are the main steps of the implementation:

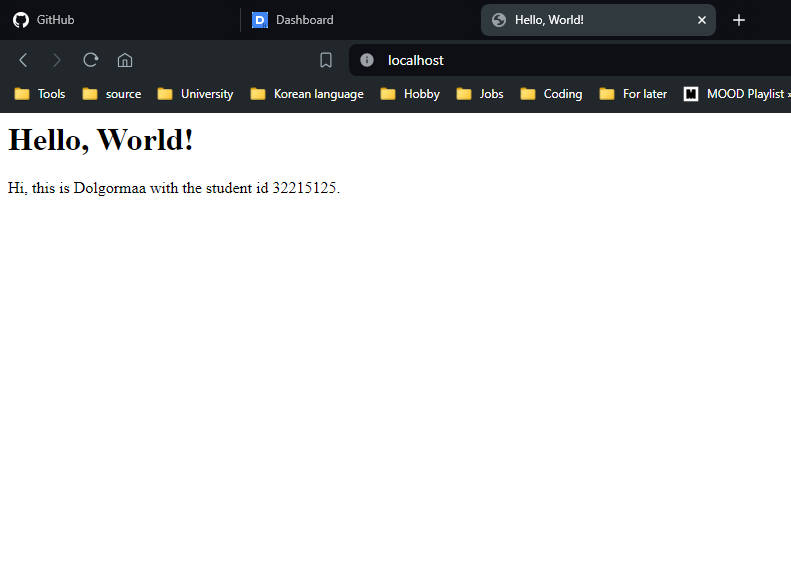
1. Opening Port 80: The server binds to localhost and port 80 using the socket.bind() method.
2. Accepting Connection Requests: The server listens for incoming connections using the socket.listen() method and accepts connections using socket.accept().
3. Reading Data from the Port: Upon accepting a connection, the server reads the incoming data from the client socket.
4. Parsing HTTP GET Request: The server parses the HTTP GET request to extract the requested file name.
5. Preparing Response: Depending on the requested file, the server prepares an appropriate HTTP response. If the requested file exists, it sends a 200 OK response along with the file content. If the file is not found, it sends a 404 Not Found response.
6. Sending Response to Client: The server sends the HTTP response back to the client using socket.send().
7. Multi-threading Support: To handle multiple clients concurrently, the server utilizes threading. Each incoming connection is handled in a separate thread, allowing the server to handle multiple requests simultaneously without blocking.

Conclusion

In conclusion, this project demonstrates the implementation of a simple web server using the socket API in Python. By following the specified requirements, the server is capable of handling basic HTTP GET requests and serving static files to clients. Additionally, the use of threading enables concurrent handling of multiple client connections, enhancing the server's scalability and performance.

Output:

Website:



Terminal:

