Web Server with Multi-threading properties

This is a C++ program that implements a basic web server. The program listens for incoming requests from clients and sends a response back to the client depending on the requested resource.

The program defines a base abstract class CClient and several derived classes Home, CFile1, CFile2, CFile3, and CExit, each of which handles a specific type of resource requested by the client.

The processClient2 method in the CClientResponse class reads the client's request and parses the requested resource. Depending on the requested resource, it creates an instance of the appropriate derived class and calls its handleClient method to generate a response for the client.

If the requested resource has been previously accessed and cached, the cached response is returned instead of generating a new response.

The program uses the winsock2.h library to handle network communication.

HTTP Background

Web browsers and web servers interact using a text-based protocol called HTTP (Hypertext Transfer Protocol). A web browser opens an Internet connection to a web server and requests some content with HTTP. The web server responds with the requested content and closes the connection. The browser reads the content and displays it on the screen.

Each piece of content on the server is associated with a file. If a client requests a specific disk file, then this is referred to as static content. If a client requests that a executable file be run and its output returned, then this is dynamic content.

Basic Web Server

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing, and delivering webpages to users.

Multi-threaded Server

Single-threaded web servers suffer from a fundamental performance problem in that only a single HTTP request can be serviced at a time. Thus, every other client that is accessing this web server must wait until the current http request has finished; this is especially a problem if the current http request is a long-running CGI program or is resident only on disk.

Error handling

This projects includes a function for handling errors related to socket programming, specifically using the Windows Sockets API. The function checks the error code returned by the WSAGetLastError() function, and throws a runtime_error exception with a corresponding error message based on the error code.

This error handling mechanism is useful in identifying and communicating errors that occur during socket programming. By providing descriptive error messages, it can help developers debug and troubleshoot their code more effectively. It also allows for more graceful handling of errors, preventing the program from crashing and providing more informative feedback to the user.

Caching

The project makes use of caching to improve the performance of the web server by reducing the number of file accesses required to serve a client request. The cache is implemented using a std::map container to store the requested resource as the key and the response as the value.

When a client requests a resource, the server first checks if the requested resource is already present in the cache. If the resource is present, the server directly returns the cached response to the client. Otherwise, the server reads the file and constructs a response, and then stores the response in the cache with the requested resource as the key.

Team Members
Ujjwal Pathak 211105
Vinayak Sharma 211117
Geetansh Singh 211418