Project Report: Python Web Server

Introduction

This project involves the development of a basic web server in Python. The goal is to create a web server capable of handling HTTP requests, serving HTML files, and implementing security features as specified in the project requirements. The project is based on a provided skeleton code and leverages Python socket programming to establish server-client communication.

Project Goals

The primary goals of this project are as follows:

- Develop a functional web server that can handle HTTP requests.
- Implement basic security measures to prevent unauthorized access to specific resources.
- Understand and apply socket programming concepts in Python.
- Test the server with various HTTP response codes (200 OK, 404 Not Found, and 403 Forbidden).

Methodology

Server Setup

- The project started with creating a server socket using Python's socket module.
- The server socket was bound to a specific address and port to listen for incoming connections.

HTTP Request Handling

- The project involved parsing HTTP requests received from clients.
- The requested file and security checks were implemented based on the project requirements.
- The leading '/' character in filenames was removed to ensure proper file access.

HTTP Response

- HTTP responses were generated and sent back to clients, including headers and file content.
- Response codes such as "200 OK," "404 Not Found," and "403 Forbidden" were handled appropriately.

Testing

- The server was tested using various scenarios:
    - Accessing an existing HTML file (resulting in "200 OK").
    - Accessing a non-existent file (resulting in "404 Not Found").
    - Attempting to access restricted resources (resulting in "403 Forbidden").

Learning Outcomes

From this project, the following key learnings were obtained:

- Socket Programming: Understanding the basics of socket programming in Python, including creating server sockets, binding them to specific addresses, and accepting client connections.
- HTTP Request Handling: Gained knowledge of parsing HTTP requests and extracting relevant information from them, such as the requested file.
- HTTP Response Generation: Learned how to construct and send HTTP responses back to clients, including response codes and headers.
- Security Measures: Implemented security checks to restrict access to specific resources, demonstrating an understanding of basic web server security.
- Troubleshooting: Developed troubleshooting skills to identify and address issues such as file not found errors and access restrictions.

Project Significance

This project is significant for the following reasons:

- Practical Application: It provides practical experience in building a basic web server, which is a fundamental component of web development.
- Understanding Web Communication: It helps in understanding the underlying mechanisms of how web servers and clients communicate over the HTTP protocol.
- Security Awareness: The project underscores the importance of implementing security measures to protect resources from unauthorized access.
- Real-world Relevance: Knowledge gained from this project is directly applicable in developing web applications and services, contributing to one's proficiency in web development.

Conclusion

In conclusion, this project enabled the development of a functional web server in Python, which can handle HTTP requests and implement basic security features. Through socket programming and HTTP request handling, this project deepened our understanding of web server fundamentals. It emphasized the importance of security and error handling in web development. Overall, this project has been a valuable learning experience in the realm of web server development and socket programming.