Realizzazione di un Web Server minimale in Python

Alessandro Gardini
alessandro.gardini7@studio.unibo.it
Matricola: 0001114867

1 giugno 2025

Obiettivo del progetto

L'obiettivo di questo progetto è realizzare un semplice server HTTP in Python, utilizzando i socket, e servire un sito web statico contenente almeno tre pagine HTML. Il server deve essere accessibile tramite `localhost:8080` e rispondere correttamente a richieste `GET` con codici di stato HTTP appropriati (`200 OK` o `404 Not Found`).

Struttura del progetto

- **server.py**: codice principale del server Python.
- **log.txt**: file di log per registrare le richieste ricevute.
- **www/**: cartella contenente i file HTML, CSS e immagini del sito web statico.

Funzionalità implementate

Gestione delle richieste HTTP

Il server gestisce le richieste HTTP di tipo `GET`, interpretando il percorso specificato e restituendo il contenuto del file corrispondente, se disponibile. In caso di richieste non valide o di errori interni, il server è in grado di rispondere con i codici di stato HTTP appropriati.

Attualmente, le risposte supportate includono:

- `200 OK` – La richiesta è stata completata con successo.
- `400 Bad Request` – La richiesta non può essere elaborata a causa di una sintassi errata.
- `404 Not Found` – La risorsa richiesta non è stata trovata.
- `405 Method Not Allowed` – Il metodo HTTP utilizzato non è supportato dal server.
- `500 Internal Server Error` – Si è verificato un errore interno al server.

MIME types

Il modulo `mimetypes` viene utilizzato per determinare il tipo di contenuto (es. `text/html`, `text/css`, `image/png`) da includere nell'intestazione HTTP in base all'estensione del file richiesto.

Logging delle richieste

Utilizzando il package `logging` di Python, le richieste vengono registrate sia nel file `log.txt` sia stampate in output standard. Vengono loggate anche altre informazioni e errori del server.

Multithreading

Il server utilizza il modulo `threading` per gestire più client contemporaneamente, creando un nuovo thread per ogni connessione in ingresso.

Layout responsive

Nel file `style.css`, incluso nel sito statico, sono state applicate regole CSS per rendere il layout adattabile ai dispositivi mobili, ad esempio ridimensionando le immagini in base alla larghezza dello schermo.

Considerazioni finali

Il server sviluppato rispetta tutti i requisiti del progetto, fornendo una base solida per eventuali estensioni future. Attualmente il server gestisce esclusivamente richieste `GET`, mentre tutte le altre vengono respinte con un messaggio di errore `405 Method Not Allowed`. L'architettura multithread e la

modularità del codice rendono semplice l'aggiunta di nuove funzionalità in futuro.

In particolare, l'utilizzo del multithreading è stato scelto anche per rendere possibile, in sviluppi successivi, l'integrazione del supporto alle connessioni persistenti tramite `keep-alive`.