

Relazione Progetto

Sviluppo di un Web Server in Python

Maddalena Prandini

Giugno 2025

1 Obiettivo del Progetto

Il progetto consiste nello sviluppo di un Web server minimale utilizzando Python, con lo scopo di fornire contenuti statici via HTTP. Il server gestisce richieste di tipo `GET`, risponde con codici di stato corretti (come 200, 403, 404, 405), e permette l'accesso a un semplice sito web ospitato localmente sulla porta 8080.

2 Organizzazione del Progetto

La struttura del progetto è la seguente:

- **www/** Contiene i file HTML e CSS del sito web da servire.
- **server.py** File principale contenente la logica del server.
- **log.txt** File di log che raccoglie le richieste gestite.

3 Funzionalità Implementate

3.1 Funzionalità Base

Il server si avvia su `localhost:8080` e gestisce connessioni in ingresso tramite socket TCP. Le funzionalità base includono:

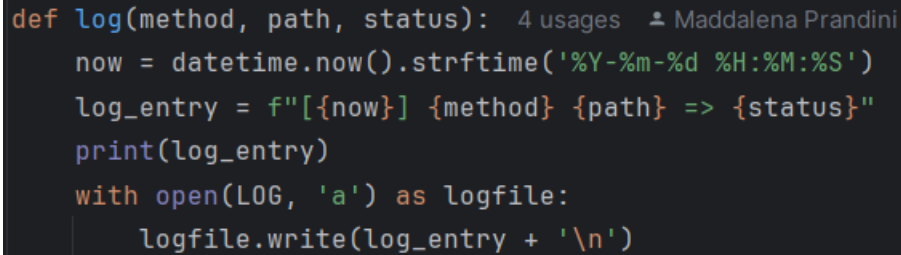
- Servizio di file statici HTML/CSS presenti nella cartella **www**.
- Risposta automatica con la homepage (`index.html`) se il path è vuoto.
- Risposta con codice 404 se la risorsa non è trovata.
- Risposta con codice 405 per metodi HTTP non supportati.

3.2 Controllo di Sicurezza

È stato implementato un controllo per impedire l'accesso a file esterni alla cartella **www**, restituendo un errore 403 in caso di violazione.

3.3 Logging

Tutte le richieste vengono registrate nel file `log.txt`, riportando data, ora, metodo HTTP, risorsa richiesta e codice di risposta. Di seguito un'immagine che riporta la funzione di log:

A screenshot of a code editor showing a Python function named `log`. The function takes three arguments: `method`, `path`, and `status`. It uses `datetime.now().strftime` to get the current time in a specific format. It then constructs a log entry string using an f-string that includes the time, method, path, and status. The function prints this entry and then opens a file named `LOG` in append mode ('a') to write the entry followed by a newline character.

```
def log(method, path, status): 4 usages  ⚙ Maddalena Prandini
    now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    log_entry = f"[{now}] {method} {path} => {status}"
    print(log_entry)
    with open(LOG, 'a') as logfile:
        logfile.write(log_entry + '\n')
```

Figura 1: Logging Function

3.4 Supporto MIME Types

Il server utilizza il modulo `mimetypes` per determinare automaticamente il tipo di contenuto da restituire, impostando correttamente l'header `Content-Type` nelle risposte.

4 Conclusioni

Il progetto ha permesso di costruire un server HTTP semplice ma funzionale, rispettando i requisiti minimi e includendo caratteristiche opzionali utili come il logging e MIME types.