Realizzazione di un Web Server minimale in `Python` e pubblicazione di un sito statico

Ettore Spaccini
Matricola: 0001117286
`ettore.spaccini@studio.unibo.it`

25 giugno 2025

Sommario

In questo documento viene descritta la progettazione e l'implementazione di un server HTTP minimale realizzato in `Python` con il modulo `socket`, incaricato di fornire un sito web statico composto da tre pagine HTML, fogli di stile CSS e risorse multimediali. Vengono inoltre discusse le estensioni facoltative sviluppate, i test di funzionamento e le possibili evoluzioni future.

Indice

1 Introduzione

Con l'avvento delle tecnologie web, comprendere i meccanismi di base che regolano la comunicazione tra *client* e *server* HTTP è fondamentale.
L'obiettivo di questo progetto è costruire un prototipo didattico che:

- implementi, a basso livello, le primitive del protocollo HTTP tramite `socket` in Python;
- renda disponibile un piccolo sito vetrina per un ipotetico ristorante;
- serva come base di partenza per successivi miglioramenti.

2 Obiettivi e Requisiti

La consegna prevedeva i seguenti requisiti minimi:

- ascolto su `localhost:8080`;
- almeno tre pagine HTML statiche;
- gestione delle richieste `GET` con risposta `200 OK`;
- risposta `404 Not Found` per risorse inesistenti.

Sono inoltre state implementate alcune estensioni opzionali:

- determinazione automatica del *MIME type* tramite il modulo `mimetypes`;
- logging a console di ogni richiesta con data, ora, metodo e codice di stato;
- layout *responsive* e leggere animazioni CSS per l'intestazione delle pagine.

3 Architettura del Sistema

3.1 Panoramica

Il progetto è suddiviso in due macro-componenti:

1. **Server HTTP** (`server.py`): si occupa di ricevere le connessioni TCP, analizzare la richiesta HTTP e restituire la risposta appropriata;
2. **Sito statico** (cartella `www/`): contiene i file HTML, CSS e le risorse multimediali.

3.2 Struttura delle Cartelle

```
Spaccini_WebServer_Ristorante/
 server.py
 www
     index.html
     menu.html
     contatti.html
```

```
css
   style.css
images
    arancini.jpg
    cannoli.jpg
    location.jpg
    ristorante.jpg
```

4 Implementazione del Server

4.1 Gestione del Socket

Il server crea un `socket` IPv4, lo associa all'host `localhost` e alla porta `8080`, quindi rimane in ascolto:

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))  # HOST = 'localhost'
    s.listen(5)
    ...
```

4.2 Parsing della Richiesta

Il metodo `handle_request` analizza la *request line*, verifica che il metodo sia `GET` e mappa l'URI alla risorsa sul disco.

4.3 Gestione delle Risposte

Se il file esiste, viene letto in binario, determinato il *Content-Type* e inviata la risposta completa di intestazioni; in caso contrario viene restituito un errore `404`:

Listing 1: Estratto di handle_request

```
if os.path.isfile(file_path):
    with open(file_path, 'rb') as f:
        body = f.read()
    content_type = mimetypes.guess_type(file_path)[0] or 'application/
       octet-stream'
    response = (
        'HTTP/1.1 200 OK\r\n'
        f'Content-Type: {content_type}\r\n'
        f'Content-Length: {len(body)}\r\n'
        '\r\n'
    ).encode() + body
else:
    response = (
        'HTTP/1.1 404 Not Found\r\n'
        'Content-Type: text/html\r\n'
        '\r\n'
        '<h1>404 Not Found</h1>'
    ).encode()
```

4.4 Logging

Ogni transazione è tracciata a console:

```
def log_request(method, path, status):
    now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(f'[{now}] {method} {path} -> {status}')
```

5 Il Sito Statistico

5.1 Pagine Principali

index.html pagina di atterraggio con *hero image* e presentazione del ristorante;

menu.html elenco dei piatti con relative immagini;

contatti.html recapiti, mappa e form (non funzionale) di contatto.

Tutte le pagine condividono lo stesso foglio di stile `style.css` in cui sono presenti:

- palette di colori caldi ispirati alla cucina mediterranea;
- `@media query` per l'adattamento su dispositivi mobili;
- animazione `slideDown` applicata all'`header`.

6 Istruzioni per l'Uso

1. Clonare (o scaricare) il repository e posizionarsi nella cartella radice:

   ```
   cd Spaccini_WebServer_Ristorante
   ```

2. Avviare il server:

   ```
   python server.py
   ```

3. Aprire il browser all'indirizzo `http://localhost:8080`.

7 Test e Risultati

Sono stati effettuati i seguenti test manuali:

- richiesta di ciascuna pagina (`/`, `/menu.html`, `/contatti.html`) con esito `200 OK`;
- richiesta di una risorsa inesistente (`/xyz.html`) con corretta risposta `404 Not Found`;
- verifica del campo `Content-Type` per HTML, CSS e immagini JPEG;
- ridimensionamento della finestra per testare la resa su schermi piccoli.

In tutti i casi il comportamento è risultato conforme alle specifiche.

8 Considerazioni Finali

Il progetto soddisfa i requisiti base ed implementa diverse estensioni opzionali. Possibili miglioramenti futuri includono:

- supporto ai metodi `POST`/`PUT`;
- gestione concorrente con `thread` o `asyncio`;
- compressione delle risposte (`gzip`);
- integrazione di template engine per generare contenuti dinamici.