Documentazione SimpleWebServer

Lorenzo Morri, Matricola: 0001077662

10 giugno 2024

1 Introduzione

Ho scelto come traccia per il progetto la numero 2: Web Server Semplice. Il SimpleWebServer consiste in un semplice script Phyton progettato per restituire file statici e gestire richieste HTTP GET di base. Per poter gestire più di una richiesta viene utilizzato il threading.

2 Codice

Il codice presentato come soluzione della traccia è il seguente:

```
#!/bin/env python
import http.server
import socketserver
import threading

#Num porta
port = 8080

# Uso il ThreadingTCPServer per gestire piu' richieste
server = socketserver.ThreadingTCPServer(('',port), http.
    server.SimpleHTTPRequestHandler )

#Termina correttamente tutti i thread
server.daemon_threads = True
#Sovrascrivo per riutilizzare socket
server.allow_reuse_address = True

#Funzione che termina processo
def stop_server():
  input("Press Enter to stop the server...")
  #Ferma serve_forever()
  server.shutdown()
```

```
22
23 # Riceve input tastiera
24 input_thread = threading.Thread(target=stop_server)
25 input_thread.daemon = True
26 input_thread.start()
27
28 try:
29   print("Serving at port", port)
30   server.serve_forever()
31 except KeyboardInterrupt:
32   pass
33 finally:
34   #Eccezioni o meno chiude tutto
35   server.server_close()
```

Listing 1: SimpleWebServer.py

Insieme a questa documentazione, nel repository del progetto è presente pure un piccolo script html e un'immagine. Di seguito lo script html:

```
1  <!DOCTYPE html>
2  <html lang="it">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
           initial-scale=1.0">
6      <title>Indice SWS</title>
7      <style>
8          body {
9              font-family: Arial, sans-serif;
10             background-color: #f0f0f0;
11             margin: 0;
12             padding: 0;
13         }
14         header {
15             background-color: #4CAF50;
16             color: white;
17             text-align: center;
18             padding: 1em 0;
19         }
20         main {
21             padding: 2em;
22             text-align: center;
23         }
24         footer {
25             background-color: #4CAF50;
26             color: white;
27             text-align: center;
28             padding: 1em 0;
29             position: fixed;
```

```
            width: 100%;
            bottom: 0;
        }
    </style>
</head>
<body>
    <header>
        <h1>index.html SimpleWebServer</h1>
    </header>
    <main>
        <p>Pagina di prova per il corretto funzionamento</p>
        <p><a href="turtle.jpg">Immagine</a></p>
        <p><a href="Documentazione.pdf">Documentazione
           progetto</a></p>
        <p><a href="SimpleWebServer.py">Codice Server</a></p>
    </main>
    <footer>
        <p>&copy; 2024 index.html</p>
    </footer>
</body>
</html>
```

Listing 2: index.html

3 Avvio Server

Per avviare il SimpleWebServer, seguire i seguenti passaggi:

1. Aprire una finestra di terminale.
2. Spostarsi nella cartella che contiene il server (`SimpleWebServer.py`) e i file da servire.
3. Eseguire poi il seguente comando:
   `python SimpleWebServer.py`
4. Una volta che il server è partito, aprire un browser web e andare su `http://localhost:8080` per accedere ai file.

Per eseguire il server in Spyder invece seguire questi passaggi:

1. Aprire Spyder.
2. Aprire script del server.

3. Fare clic sull'icona "Esegui" (una freccia verde) o premere il tasto F5 per eseguire il codice.

4. Controllare l'output nella console di Spyder, e in caso di successo aprire un browser web e andare su `http://localhost:8080` per accedere ai file.

In automatico il server aprirà la pagina index.html.

4 Considerazioni Aggiuntive

- **Posizione dei File**: I file si devono trovare nella stessa cartella del server. Il server servirà i file da questa cartella e in caso dalle sue sottocartelle.

- **Terminazione del Server**: Il server sarà in esecuzione finché non verrà premuto il tasto "Invio" nella finestra del terminale in cui è in esecuzione (cmd o console Spyder). Una volta premuto, il server si fermerà e lascerà la porta che stava ascoltando.

- **Estendibilità del Server**: Il Server può essere esteso per essere personalizzato a proprio piacimento. Si possono implementare nuove funzionalità come ad esempio le richieste POST o un login per gli utenti. Questa flessibilità consente l'adattamento per la getsione delle proprie esigenze.

- **Protocollo HTTP**: Il protocollo utilizzato è quello HTTP. I codici di stato utlizzati principalmente dal server sono 200 OK e 404 Not Found. Per renderlo più sicuro è consigliato utilizzare il protocollo HTTPS.

- **Gestione delle Prestazioni**: Oltre alla gestione delle richieste HTTP, è fondamentale anche considerare le prestazioni. Ottimizzazioni come la memorizzazione, la riduzione della dimensione delle risposte e l'implementazione di una gestione della concorrenza possono migliorare le prestazioni del server.