TASK 03

PROGRAMMA IN PYTHON PER L'ENUMERAZIONE DEI METODI HTTP

A CURA DI: EDOARDO MUDADU, ELENA KOVALENKO

INTRODUZIONE:

Questo codice scritto in linguaggio Python svolge la funzione di eseguire una serie di richieste http con diversi verbi a un URL specificato, e successivamente determinare quali di questi verbi sono abilitati dal server. L'utente è invitato ad inserire un URL, ed il programma invierà una richiesta http utilizzando i vari verbi per controllare lo stato di risposta del server per ciascun verbo.
Infine, verrà stampato l'elenco dei verbi http che hanno restituito uno stato di risposta inferiore a 400 (considerabili "abilitati").

STRUTTURA:

LA LIBRERIA: "requests"; questa libreria in Python semplifica l'invio di richieste http. Con questa libreria possiamo realizzare richieste http complesse con poche linee di codice, riducendo la complessità rispetto all'utilizzo di librerie più basse o alle chiamate socket dirette.
FUNZIONE "controlla_verbi_http": la funzione prende un parametro 'URL' come input e itera attraverso una lista predefinita di verbi http. Per ciascun verbo, invia una richiesta http all'URL specificato e verifica lo stato della risposta.
Come detto prima, se lo stato è inferiore a 400, il verbo è considerato abilitato e registrato in una lista.

GESTIONE DEGLI ERRORI: la funzione gestisce eventuali eccezioni che si potrebbero verificare durante l'invio delle richieste, ad esempio un errore di connessione.
In tal caso verrà stampato un messaggio di errore specifico.
INPUT DALL'UTENTE: l'utente inserisce l'URL attraverso l'input.
Questo URL sarà poi utilizzato per eseguire la verifica dei verbi http.
STAMPA DEI RISULTATI: Alla fine del programma, viene stampato l'elenco dei verbi http abilitati per l'URL specificato.

Codice complessivo di commenti:

```
import requests
#Importiamo la libreria per effettuare richieste HTTP

def controlla_verbi_http(url):
#Definisce una funzione che accetta un parametro 'url'.

    verbi_http = ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD', 'PATCH', 'CONNECT']
    #Creata una lista che contenente i principali verbi HTTP.

    verbi_abilitati = []
   #Inizializza una lista vuota di verbi che verrà utilizzata per memorizzare i verbi HTTP supportati dall'URL.

    for verbo in verbi_http:
    #Entra in un ciclo for che itera attraverso ogni verbo HTTP nella lista verbi_http.
        try:

            response = requests.request(verbo, url)
  #All'interno del ciclo, prova a effettuare una richiesta HTTP usando il verbo corrente e l'URL fornito.
            if response.status_code < 400:

#Se lo status code inferiore a 400), aggiunge il verbo alla  lista verbi_abilitati e stampa il verbo HTTP e il relativo status code.

                verbi_abilitati.append(verbo)
            print(f"Verbo HTTP: {verbo}")
            print(f"Status code: {response.status_code}")

        except requests.exceptions.RequestException as e:
        #Se si verifica un'eccezione durante la richiesta

            print(f"Errore durante {verbo}: {e}")
            #viene stampato un messaggio di errore.

    print(f"I verbi HTTP abilitati per {url} sono: {verbi_abilitati}")
    #Viene stampata la lista dei verbi HTTP abilitati per un percorso specifico.

path = input("\nInserisci URL: ")
#Richiede all'utente di inserire un URL tramite input.

verbi_HTTP_abilitati = (path)
controlla_verbi_http(path)
#Chiama la funzione controlla_verbi_http passando l'URL inserito come parametro.
```

Vediamo cosa fa ciascuno dei verbi e come viene utilizzato

GET: È utilizzato per richiedere dati da una risorsa specificata e non dovrebbe avere effetti collaterali sul server. È abilitato perché è comunemente usato per ottenere informazioni da un server.

POST: Viene utilizzato per inviare dati al server per creare una nuova risorsa. È abilitato perché consente l'invio di dati e la creazione di nuove risorse.

PUT*:* Serve per aggiornare o sostituire una risorsa esistente sul server. È abilitato per consentire l'aggiornamento di risorse esistenti.

DELETE*:* Utilizzato per eliminare una risorsa specificata dal server. È abilitato per consentire la rimozione di risorse.

OPTIONS*:* Viene utilizzato per ottenere le opzioni di comunicazione disponibili per una risorsa o il server stesso. È spesso abilitato per fornire informazioni sulle operazioni supportate dal server.

HEAD*:* Simile a GET, ma richiede solo l'intestazione della risorsa senza il corpo della risposta. È utile per ottenere metadati di una risorsa senza scaricare tutto il contenuto. È abilitato perché fornisce informazioni sull'intestazione della risorsa.

PATCH*:* Viene utilizzato per applicare modifiche parziali a una risorsa. È abilitato per consentire modifiche parziali senza richiedere la sostituzione dell'intera risorsa.

CONNECT: Usato per richiedere un tunnel TCP verso il server identificato dal resource URI. È più comunemente utilizzato in situazioni di proxying. Potrebbe essere disabilitato per ragioni di sicurezza, in quanto apre la possibilità di connessioni a server esterni attraverso un proxy.

Proviamo ora a fare il test utilizzando il nostro programma

sul server "http://192.168.50.101/phpMyAdmin"

Controllo dei verbi abilitati sul server "http://192.168.50.101/phpMyAdmin"

Possiamo vedere che tutti i verbi tranne CONNECT sembrano essere abilitati, il che potrebbe indicare che il server consente operazioni di lettura, scrittura, aggiornamento e eliminazione delle risorse.
Il verbo CONNECT potrebbe essere disabilitato per ragioni di sicurezza, poiché aprirlo potrebbe permettere connessioni a server esterni attraverso un proxy, il che potrebbe essere indesiderato in certi contesti di sicurezza delle reti.

Quindi possiamo concludere che:

Quando si lavora con servizi Web che rispettano il protocollo HTTP, è fondamentale utilizzare i metodi appropriati per interagire con le risorse remote.

Se si creano servizi Web in Python, l'enumerazione dei metodi HTTP aiuta a definire le azioni che un client può eseguire su determinate risorse, facilitando la costruzione di API ben definite.

In sostanza, l'enumerazione dei metodi HTTP in Python è fondamentale per comunicare con servizi Web, definire operazioni su risorse remote e costruire o utilizzare API RESTful (utilizzate per sviluppare servizi web che forniscono accesso a dati e funzionalità tramite richieste HTTP.) in modo efficace. Utilizzando librerie come requests, è possibile eseguire richieste con vari metodi HTTP per svolgere azioni specifiche su server remoti.