Python: usare i WebSocket

I WebSocket sono un protocollo di comunicazione bidirezionale che consente agli sviluppatori di creare applicazioni web interattive in tempo reale. A differenza del classico modello HTTP, che è di natura stateless (senza stato), i WebSocket mantengono una connessione aperta tra il client e il server, consentendo una comunicazione più efficiente e istantanea. In questo articolo, esploreremo come utilizzare i WebSocket in Python per migliorare l'interattività delle applicazioni web.

Prerequisiti

Prima di iniziare, assicuriamoci di avere Python installato sul nostro sistema. Inoltre, installeremo la libreria `websockets` utilizzando pip:

```
pip install websockets
```

Creare un Server WebSocket

Iniziamo creando un server WebSocket di base utilizzando la libreria `websockets`. Creiamo un file chiamato `websocket_server.py` e inseriamo il seguente codice:

```
import asyncio
import websockets

async def handler(websocket, path):
    # Il parametro 'websocket' rappresenta la
connessione con il client.
```

```
    # Il parametro 'path' rappresenta il percorso
della richiesta, ma non lo useremo in questo esempio.

    while True:
        # Attendiamo i messaggi dal client
        message = await websocket.recv()

        # Elaboriamo il messaggio (aggiungiamo un
prefisso)
        response = f"Server: {message}"

        # Inviamo la risposta al client
        await websocket.send(response)

# Configuriamo il server WebSocket
start_server = websockets.serve(handler, "localhost",
8765)

# Avviamo il server
asyncio.get_event_loop().run_until_complete(start_ser
ver)
asyncio.get_event_loop().run_forever()
```

Il codice sopra definisce un server WebSocket di base che ascolta su `localhost` alla porta `8765`. Ogni volta che riceve un messaggio dal client, risponde aggiungendo un prefisso "Server:".

Creare un Client WebSocket

Ora, creiamo un client WebSocket di base. Creiamo un file chiamato `websocket_client.py` e inseriamo il seguente codice:

```
import asyncio
```

```
import websockets

async def hello():
    uri = "ws://localhost:8765"

    async with websockets.connect(uri) as websocket:
        # Inviamo un messaggio al server
        message = "Ciao, server!"
        await websocket.send(message)
        print(f"Inviato: {message}")

        # Riceviamo la risposta dal server
        response = await websocket.recv()
        print(f'Risposta ricevuta: {response}')

# Eseguiamo il client WebSocket
asyncio.get_event_loop().run_until_complete(hello())
```

Il codice sopra definisce un client WebSocket che si connette al server su `localhost` alla porta `8765`. Invia un messaggio al server, riceve la risposta e la stampa a console.

Eseguire Server e Client

Per eseguire il server, apriamo un terminale e eseguiamo il seguente comando:

```
python websocket_server.py
```

Successivamente, apriamo un altro terminale e eseguiamo il client:

```
python websocket_client.py
```

Osserviamo l'output nei terminali per vedere l'interazione tra server e client attraverso la connessione WebSocket.

Conclusioni

L'utilizzo dei WebSocket in Python consente di implementare comunicazioni bidirezionali in tempo reale tra server e client. La libreria `websockets` semplifica l'implementazione di server e client WebSocket, consentendo agli sviluppatori di creare applicazioni web più interattive e dinamiche. Sperimenta ulteriormente con questa tecnologia per integrare funzionalità in tempo reale nelle tue applicazioni web Python.