Go: using WebSockets

WebSockets are a two-way communication protocol over a single TCP connection, designed to provide real-time communication between clients and servers over the web. This technology is widely used in modern web applications that require instant communication, such as live chats, online games, push notifications, and more.

In this article, well explore how to implement WebSockets in Go, a programming language that enjoys high performance and ease of use.

Go provides a `net/http` package for handling HTTP requests, but to implement WebSockets, we will be using the third-party library called "Gorilla WebSockets". This library greatly simplifies the implementation of WebSockets in Go.

To install Gorilla WebSocket, run the following command:

```
go get github.com/gorilla/websocket
```

Now we can start writing code for our WebSocket server. Lets create a file called `server.go` in our project directory and start implementing the server:

```
// server.go
package main

import (
    "log"
    "net/http"
```

```
    "github.com/gorilla/websocket"
)

// Creiamo un upgrader per aggiornare la connessione
HTTP a WebSocket
var upgrader = websocket.Upgrader{
    CheckOrigin: func(r *http.Request) bool {
        return true
    },
}

func handleWebSocket(w http.ResponseWriter, r
*http.Request) {
    // Aggiorniamo la connessione HTTP a WebSocket
    conn, err := upgrader.Upgrade(w, r, nil)
    if err != nil {
        log.Println("Errore durante l'aggiornamento
della connessione HTTP a WebSocket:", err)
        return
    }
    defer conn.Close()

    for {
        // Leggiamo il messaggio ricevuto dal client
        messageType, message, err :=
conn.ReadMessage()
        if err != nil {
            log.Println("Errore durante la lettura
del messaggio:", err)
            break
        }

        // Stampiamo il messaggio ricevuto dal client
sulla console del server
```

```
        log.Printf("Messaggio ricevuto: %s", message)

        // Inviamo un messaggio di risposta al client
con lo stesso messaggio ricevuto
        err = conn.WriteMessage(messageType, message)
        if err != nil {
            log.Println("Errore durante l'invio del
messaggio di risposta:", err)
            break
        }
    }
}

func main() {
    http.HandleFunc("/ws", handleWebSocket)
    log.Println("Server WebSocket in ascolto su
http://localhost:8080/ws")
    err := http.ListenAndServe(":8080", nil)
    if err != nil {
        log.Fatal("Errore durante l'avvio del
server:", err)
    }
}
```

Now that we have our WebSocket server implemented, we can compile and run it:

```
go build
./websocket-example
```

The WebSocket server is now listening at `http://localhost:8080/ws`. You can connect to the WebSocket server using a WebSocket client library or a WebSocket testing tool such as "websocat" or "websocket.org".