GABRIELE ROMANATO

Gestione della sessione in Go

La gestione della sessione in Go non è integrata nel linguaggio o nel pacchetto standard per applicazioni web. Tuttavia, esistono librerie che semplificano questo compito, come `gorilla/sessions`. In questo articolo vedremo come implementare una gestione base della sessione utilizzando questa libreria.

Installazione della libreria

Per prima cosa, è necessario installare il pacchetto `gorilla/sessions`:

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

Creazione del gestore di sessione

Il seguente esempio mostra come configurare e utilizzare una sessione:

```
package main

import (
    "fmt"
    "net/http"
    "github.com/gorilla/sessions"
)

var store = sessions.NewCookieStore([]byte("super-secret-
key"))

func loginHandler(w http.ResponseWriter, r *http.Request) {
    session, _ := store.Get(r, "session-name")
    session.Values["authenticated"] = true
    session.Save(r, w)
    fmt.Fprintln(w, "Logged in")
```

```
}

func logoutHandler(w http.ResponseWriter, r *http.Request) {
    session, _ := store.Get(r, "session-name")
    session.Values["authenticated"] = false
    session.Save(r, w)
    fmt.Fprintln(w, "Logged out")
}

func protectedHandler(w http.ResponseWriter, r *http.Request)
{
    session, _ := store.Get(r, "session-name")
    if auth, ok := session.Values["authenticated"].(bool);
!ok || !auth {
        http.Error(w, "Forbidden", http.StatusForbidden)
        return
    }
    fmt.Fprintln(w, "This is a protected area")
}

func main() {
    http.HandleFunc("/login", loginHandler)
    http.HandleFunc("/logout", logoutHandler)
    http.HandleFunc("/protected", protectedHandler)
    http.ListenAndServe(":8080", nil)
}
```

Spiegazione del codice

- **store**: viene inizializzato con una chiave segreta, usata per firmare i cookie.
- **loginHandler**: imposta il valore `authenticated` a `true`.
- **logoutHandler**: imposta il valore `authenticated` a `false`.
- **protectedHandler**: verifica se l'utente è autenticato.

Considerazioni di sicurezza

È importante utilizzare una chiave segreta forte e unica. In produzione, considera l'utilizzo di store di sessione alternativi, come Redis o database, per una maggiore scalabilità e sicurezza.