Implementare l'upload di file multipli con Go

In questo articolo, vedremo come implementare l'upload di file multipli in un'applicazione web sviluppata con il linguaggio Go. Questo approccio è utile per gestire l'upload di più file contemporaneamente, ad esempio immagini o documenti, da parte degli utenti.

Implementazione

Creiamo un'applicazione Go che permette agli utenti di caricare più file.

1. Configurare il Server HTTP

Creiamo un file `main.go` e configuriamo un server HTTP per gestire le richieste di upload.

```
package main

import (
        "fmt"
        "io"
        "net/http"
        "os"
)

func main() {
        http.HandleFunc("/upload", uploadHandler)
        http.HandleFunc("/", formHandler)

        fmt.Println("Server in ascolto sulla porta 8080...")
        http.ListenAndServe(":8080", nil)
}
```

2. Creare il Form HTML

Il modulo HTML consente agli utenti di selezionare più file e inviarli al server.

```
func formHandler(w http.ResponseWriter, r *http.Request) {
        html := `
        <!DOCTYPE html>
        <html lang="en">
        <head>
                <meta charset="UTF-8">
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <title>Upload File Multipli</title>
        </head>
        <body>
                <h1>Carica i tuoi file</h1>
                <form action="/upload" method="post" enctype="multipart/form-data">
                        <input type="file" name="files" multiple>
                        <br><br>
                        <button type="submit">Carica</button>
                </form>
        </body>
        </html>
        `
```

```
        w.Write([]byte(html))
}
```

3. Gestire l'Upload dei File

Creiamo un handler che processa i file inviati dal modulo HTML.

```
func uploadHandler(w http.ResponseWriter, r *http.Request) {
        if r.Method != http.MethodPost {
                http.Error(w, "Metodo non consentito", http.StatusMethodNotAllowed)
                return
        }

        r.ParseMultipartForm(10 << 20) // Limite di 10 MB

        files := r.MultipartForm.File["files"]
        for _, fileHeader := range files {
                file, err := fileHeader.Open()
                if err != nil {
                        http.Error(w, "Errore durante l'apertura del file",
http.StatusInternalServerError)
                        return
                }
                defer file.Close()

                out, err := os.Create("uploads/" + fileHeader.Filename)
                if err != nil {
                        http.Error(w, "Errore durante la creazione del file",
http.StatusInternalServerError)
                        return
                }
                defer out.Close()

                _, err = io.Copy(out, file)
                if err != nil {
                        http.Error(w, "Errore durante la scrittura del file",
http.StatusInternalServerError)
                        return
                }
        }

        w.Write([]byte("Upload completato!"))
}
```

Testare l'Applicazione

Avviare l'applicazione eseguendo il comando:

```
go run main.go
```

Accedere al modulo di upload all'indirizzo `http://localhost:8080` e testare l'upload di più file.

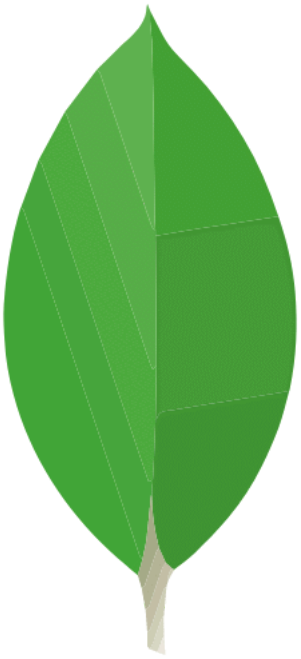
Conclusione

Abbiamo visto come implementare l'upload di file multipli con Go utilizzando un modulo HTML e un server HTTP. Questo approccio può essere personalizzato per soddisfare le esigenze specifiche della tua applicazione.

Applicazioni Correlate

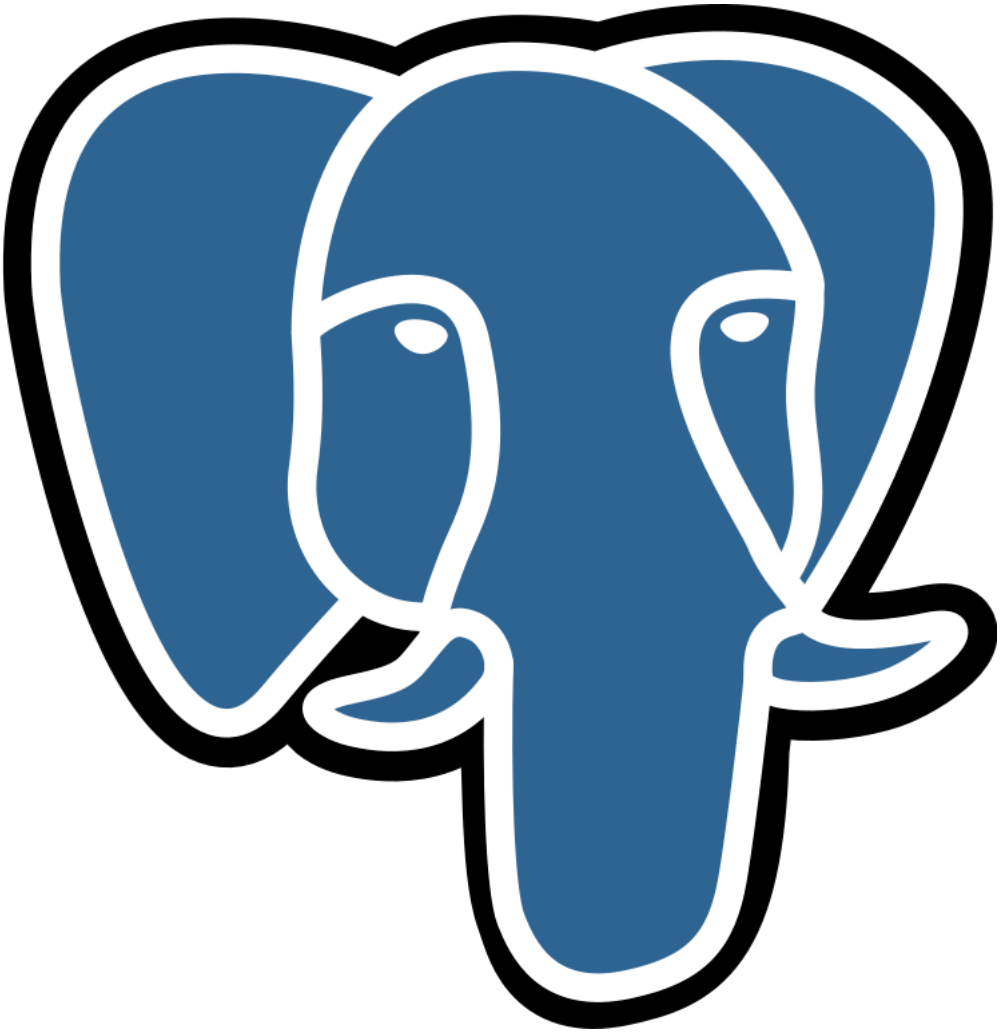
- ### Go Placeholder Image

  Applicazione in Go per la creazione di immagini segnaposto.
  DockerDocker ComposeGoJavaScript



Go MongoDB App

Applicazione basata su MongoDB ed implementata in Go con il driver ufficiale.
DockerDocker ComposeGoMongoDB



Go PostgreSQL App

Applicazione basata su PostgreSQL e sviluppata in Go.
DockerDocker ComposeGoPostgreSQL