Lección 3 - Crear un servidor web

Tabla de contenido

Introducción

En esta clase, nos sumergiremos en la creación de un servidor web utilizando el Framework Gorilla de Go. Gorilla nos ofrece un enfoque modular para definir rutas y manejar solicitudes, lo que facilita la construcción de aplicaciones web escalables y mantenibles.

Objetivos

1. **Configuración de Rutas y Manejo de Solicitudes:**
   - Aprender a utilizar las capacidades de Gorilla para definir rutas y manejar solicitudes HTTP de manera eficiente.
2. **Implementación de Middleware:**
   - Explorar cómo agregar middleware a nuestras rutas para realizar operaciones adicionales, como registro de solicitudes o manejo de autenticación.
3. **Creación de una Ruta Dinámica:**
   - Implementar una ruta que acepte parámetros dinámicos en la URL y explore cómo acceder a estos parámetros desde el código.

Desarrollo

Configuración de Rutas y Manejo de Solicitudes

- Utilizaremos el paquete `mux` de Gorilla para definir rutas y manejar solicitudes de manera clara y estructurada. Ejemplo básico:

```
package main

import (
    "fmt"
    "log"
    "net/http"

    "github.com/gorilla/mux"
)

func main() {
    r := mux.NewRouter()

    // Ruta principal
    r.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprint(w, "¡Bienvenido al servidor Gorilla!")
    })

    // Iniciar el servidor en el puerto 8080
    log.Println("Servidor Gorilla en ejecución en http://localhost:8080")
    http.ListenAndServe(":8080", r)
}
```

Implementación de Middleware

- Agregaremos middleware a nuestras rutas para realizar operaciones antes o después de manejar la solicitud. Ejemplo básico de middleware de registro:

```
// Aplicar middleware de registro a todas las solicitudes
r.Use(func(next http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r
*http.Request) {
                log.Println(r.Method, r.URL.Path)
                next.ServeHTTP(w, r)
        })
})
```

Agregar la llamada a la función en `main.go`.

```
// Agregar middleware de registro
r.Use(LoggingMiddleware)
```

```
// LoggingMiddleware registra el método de solicitud, la URL, el código de estado y la latencia
func LoggingMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        start := time.Now()

        // Llama al siguiente controlador en la cadena de middleware
        next.ServeHTTP(w, r)

        // Calcula la latencia
        latency := time.Since(start)

        // Registra el método de solicitud, la URL, el código de estado y la latencia
        log.Printf("%s %s - %v - %s", r.Method, r.URL.Path, latency, http.StatusText(http.StatusOK))
    })
}
```

Creación de una Ruta Dinámica

- Implementaremos una ruta que acepte parámetros dinámicos en la URL y utilice estos parámetros en la lógica de manejo. Ejemplo:

```
// Definir una ruta con una variable de ruta
r.HandleFunc("/saludo/{nombre}", func(w http.ResponseWriter, r *http.Request) {
        vars := mux.Vars(r)
        nombre := vars["nombre"]
        w.Write([]byte("¡Hola, " + nombre + "!"))
})
```

Este código define una ruta `/saludo/{nombre}` que responde con un saludo personalizado basado en el nombre proporcionado en la URL. También aplica un middleware de registro a todas las solicitudes para registrar información sobre cada solicitud en la terminal.

Conclusión

En esta lección, hemos aprendido cómo crear un servidor web básico utilizando el framework Gorilla en Go. A través de los objetivos establecidos, hemos logrado:

1. **Configurar un Servidor Básico con Gorilla:**
   - Inicializamos un servidor web básico y lo configuramos para escuchar en el puerto 8080.
2. **Responder a Solicitudes HTTP:**
   - Respondimos a solicitudes HTTP utilizando Gorilla, enviando mensajes de texto plano y JSON como respuestas.
3. **Manejar Enrutamiento y Middleware:**
   - Definimos rutas con variables de ruta y aplicamos middleware para tareas como registro de solicitudes.

Hemos adquirido una comprensión sólida de cómo utilizar Gorilla para construir servidores web en Go y estamos preparados para explorar características más avanzadas del framework en futuras lecciones.

Conoce más del autor

¡Encuéntrame en las siguientes redes sociales para estar al tanto de mis proyectos y actividades!

Red Social	Enlace
Página web	jersonmartinez.com
LinkedIn	Jerson Martínez - DevOps Engineer
Canales de YouTube	DevOpsea | Side Master
GitHub	Perfil en GitHub
Twitter (X)	@antoniomorenosm