Lección 4 - Crear rutas y manejar peticiones

Tabla de contenido

Introducción

En esta clase, nos enfocaremos en crear rutas y controladores utilizando las funciones proporcionadas por Gin. Comenzaremos definiendo rutas básicas y luego avanzaremos hacia rutas más dinámicas que acepten parámetros. Además, aprenderemos a separar el manejo de rutas en un archivo independiente para mantener nuestro código organizado y modular.

Objetivos

1. **Definir rutas y controladores básicos**: Empezaremos por definir rutas estáticas en el archivo `main.go` utilizando las funciones `GET` de Gin. Esto nos permitirá establecer el flujo básico de nuestra aplicación y responder a las solicitudes HTTP de manera simple.
2. **Manejar rutas dinámicas con parámetros**: Avanzaremos hacia la definición de rutas dinámicas que acepten parámetros en la URL. Aprenderemos a extraer estos parámetros y utilizarlos en nuestros controladores para generar respuestas personalizadas basadas en la entrada del usuario.
3. **Separar el manejo de rutas en un archivo independiente**: Para mejorar la organización y mantenibilidad de nuestro código, moveremos el manejo de rutas a un archivo separado llamado `routes/routes.go`. Esto nos permitirá tener un mejor control sobre nuestras rutas y facilitará la expansión de nuestra aplicación en el futuro.

Desarrollo

Crear Rutas y Manejar Peticiones

- En el archivo `main.go`, defina rutas y controladores utilizando las funciones proporcionadas por Gin. Por ejemplo:

```
package main

// Se importan los paquetes "gin" y "net/http".
import (
	// "gin" es el paquete principal que proporciona el framework web,
	"github.com/gin-gonic/gin"

	// mientras que "net/http" se utiliza para manipular solicitudes y
respuestas HTTP.
	"net/http"
)

func main() {
	//Se crea una nueva instancia del motor Gin utilizando la función
`gin.Default()`, que configura el enrutador con el middleware de registro
y recuperación por defecto.
	r := gin.Default()

	// Ruta básica
	r.GET("/", func(c *gin.Context) {
		c.String(http.StatusOK, "¡Hola, mundo!")
	})
	// Se define una ruta para la URL raíz ("/") que responde con el
mensaje "¡Hola, mundo!" con un código de estado HTTP 200 (OK) utilizando
`c.String()`.

	// Ruta con parámetros
	r.GET("/saludo/:nombre", func(c *gin.Context) {
		nombre := c.Param("nombre")
		c.String(http.StatusOK, "¡Hola, "+nombre+"!")
	})
	// Se define una ruta para "/saludo/:nombre" que espera un
parámetro "nombre" en la URL. El nombre se recupera utilizando
`c.Param("nombre")` y se utiliza para construir un saludo personalizado
que se devuelve como respuesta.

	r.Run(":8080")
	// Finalmente, el servidor se inicia en el puerto 8080 utilizando
```

```
`r.Run(":8080")`.
}
```

Separar el manejo de rutas

Trabajaremos sobre el fichero `routes/routes.go`.

```
package routes

import (
        "net/http"
        "github.com/gin-gonic/gin"
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(r *gin.Engine) {
        // Ruta básica
        r.GET("/", func(c *gin.Context) {
                // Handler para la ruta raíz "/"
                c.String(http.StatusOK, "¡Hola, mundo!") // Responde con
un mensaje de saludo
        })

        // Ruta con parámetros
        r.GET("/saludo/:nombre", func(c *gin.Context) {
                // Handler para la ruta "/saludo/:nombre"
                nombre := c.Param("nombre") // Obtiene el parámetro
"nombre" de la URL
                c.String(http.StatusOK, "!Hola, "+nombre+"!") // Responde
con un mensaje de saludo personalizado.
        })
}
```

Modificamos el fichero `main.go`:

```
package main

import (
    "Gin/routes"
    "github.com/gin-gonic/gin"
)
```

```
func main() {
    r := gin.Default()

    routes.SetupRoutes(r)

    r.Run(":8080")
}
```

```
$ go run main.go
```

Probar en el navegador:

```
127.0.0.1:8080/saludo/Jerson
```

Estos comentarios explican qué hace cada parte del código:

- `SetupRoutes`: Esta función configura las rutas de la aplicación. Recibe un puntero a un `gin.Engine`, que es el motor de enrutamiento de Gin.
- `r.GET("/", func(c *gin.Context) { ... })`: Define una ruta para la URL raíz ("/"). Cuando se accede a esta ruta mediante una solicitud GET, se ejecuta la función de controlador anónima que devuelve un mensaje de saludo.
- `c.String(http.StatusOK, "¡Hola, mundo!")`: Dentro del controlador de la ruta raíz, se utiliza `c.String()` para enviar una respuesta HTTP con el estado OK (200) y el mensaje "¡Hola, mundo!".
- `r.GET("/saludo/:nombre", func(c *gin.Context) { ... })`: Define una ruta que espera un parámetro "nombre" en la URL. Cuando se accede a esta ruta mediante una solicitud GET, se ejecuta la función de controlador anónima que devuelve un saludo personalizado utilizando el parámetro "nombre".
- `nombre := c.Param("nombre")`: Dentro del controlador de la ruta "/saludo/:nombre", se utiliza `c.Param()` para obtener el valor del parámetro "nombre" de la URL.
- `c.String(http.StatusOK, "!Hola, "+nombre+"!")`: Dentro del controlador de la ruta "/saludo/:nombre", se utiliza `c.String()` para enviar una respuesta HTTP con el estado OK (200) y un saludo personalizado que incluye el nombre proporcionado en la URL.

Subir los cambios a GitHub:

```
$ git add .
$ git commit -m "Crear rutas y manejar peticiones"
```

```
$ git push
```

Revisar en la web de GitHub que las actualizaciones estén disponibles. Detallaré mayor contenido sobre cada lección.

Conclusión

Hemos aprendido cómo crear rutas y manejar peticiones utilizando Gin en Go. Comenzamos con conceptos básicos como rutas estáticas y avanzamos hacia rutas dinámicas que aceptan parámetros. Además, hemos explorado cómo separar el manejo de rutas en archivos independientes para mejorar la organización de nuestro código. Con estos conocimientos, estamos preparados para desarrollar aplicaciones web más complejas y escalables en Go utilizando el poderoso framework Gin.

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