Lección 3 - Crear un servidor web

Tabla de contenido

- Introducción
- Objetivos
- Desarrollo
  - Iniciar un Servidor Echo Básico
  - Devolviendo una salida desde el servidor
  - Retornar una Respuesta JSON
  - Crear una Ruta Simple y Manejar Solicitudes
  - Hacer un logger específico
- Conclusión
- Conoce más del autor

Introducción

En esta lección, exploraremos el framework Echo de Go, aprendiendo a configurar y utilizarlo para crear un servidor web básico que responde a solicitudes HTTP. Desde establecer la configuración inicial del servidor hasta manejar diferentes tipos de solicitudes y utilizar middleware para registrar la actividad del servidor en la terminal, además de ampliar nuestra comprensión de las capacidades y la versatilidad de Echo en el desarrollo de aplicaciones web con Go, brindando una visión sólida de cómo aprovechar esta herramienta para proyectos futuros.

Objetivos

1. **Configurar un Servidor Básico con Echo:**
   - Este objetivo se centra en establecer un servidor web básico utilizando el framework Echo de Go. A través del código proporcionado, aprenderemos a inicializar un servidor Echo, configurar el puerto de escucha y poner en marcha la aplicación.
2. **Responder con Mensajes desde el Servidor:**
   - En este objetivo, nos centraremos en devolver mensajes simples desde el servidor Echo en respuesta a las solicitudes HTTP. Utilizando la función `c.String` y `c.JSON`, aprenderemos a enviar respuestas de texto plano y JSON respectivamente, para proporcionar información al cliente.

3. **Crear Rutas y Manejar Solicitudes:**
    - Aquí exploraremos cómo definir rutas adicionales en nuestra aplicación Echo para manejar diferentes tipos de solicitudes. A través del ejemplo proporcionado, aprenderemos a crear rutas simples y a responder con mensajes específicos para cada ruta definida.

Desarrollo

Iniciar un Servidor Echo Básico

Vamos a trabajar sobre el fichero `main.go`:

```
package main

import (
        "github.com/labstack/echo/v4"
)

func main() {
        // Crear una instancia de Echo
        e := echo.New()

        // Configurar el puerto de escucha
        e.Server.Addr = ":8080"

        // Iniciar la aplicación Echo
        e.StartServer(e.Server)
}
```

- Esta configuración inicial establece el puerto de escucha en `:8080`, pero puedes ajustarlo según tus preferencias.

Devolviendo una salida desde el servidor

- En tu archivo `main.go`, inicia un servidor Echo básico:

```
package main

import (
        "github.com/labstack/echo/v4"
```

```
        "net/http"
)

func main() {
        // Crear una instancia de Echo
        e := echo.New()

        // Ruta básica que responde con un mensaje
        e.GET("/", func(c echo.Context) error {
                return c.String(http.StatusOK, "¡Hola, mundo!")
        })

        // Iniciar el servidor en el puerto 8080
        e.Start(":8080")
}
```

Retornar una Respuesta JSON

Echo facilita el retorno de respuestas JSON al cliente. Por ejemplo, podemos modificar la ruta raíz para devolver un JSON en lugar de un mensaje de texto plano.

```
e.GET("/", func(c echo.Context) error {
    return c.JSON(http.StatusOK, map[string]string{
        "message": "¡Hola desde Echo!",
    })
})
```

Crear una Ruta Simple y Manejar Solicitudes

- Añade una ruta que maneje solicitudes y responda con un mensaje:

```
// ...

func main() {
        // Crear una instancia de Echo
        e := echo.New()

        // Ruta básica que responde con un mensaje
        e.GET("/", func(c echo.Context) error {
                return c.String(http.StatusOK, "¡Hola, mundo!")
        })
```

```
        // Ruta adicional para manejar otras solicitudes
        e.GET("/saludo", func(c echo.Context) error {
                return c.String(http.StatusOK, "Saludos desde Echo!")
        })

        // Iniciar el servidor en el puerto 8080
        e.Start(":8080")
}
```

Para mostrar las solicitudes en la terminal mientras ejecutas un servidor Echo, necesitas agregar un middleware de registro a tu aplicación Echo. Aquí te muestro cómo hacerlo:

Primero, necesitas instalar la dependencia del middleware de registro. Ejecuta el siguiente comando en tu terminal:

```
$ go get github.com/labstack/echo/v4/middleware@v4.11.4
```

```
package main

import (
        "github.com/labstack/echo/v4"
        "github.com/labstack/echo/v4/middleware"
)

func main() {
        // Crea una nueva instancia de Echo
        e := echo.New()

        // Agrega el middleware de registro para registrar las solicitudes
en la terminal
        e.Use(middleware.Logger())

        // Define una ruta de ejemplo
        e.GET("/", func(c echo.Context) error {
                return c.String(http.StatusOK, "¡Hola, desde Echo!")
        })

        // Inicia el servidor en el puerto 8080
        e.Start(":8080")
}
```

Hacer un logger específico

```
// Middleware de registro de solicitudes
e.Use(middleware.LoggerWithConfig(middleware.LoggerConfig{
        Format: "method=${method}, uri=${uri}, status=${status},
time=${latency_human}\n",
}))
```

Subir los cambios a GitHub:

```
$ git add .
$ git commit -m "Crear un servidr web"
$ git push
```

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

Conclusión

En esta lección, hemos explorado cómo utilizar el framework Echo de Go para crear un servidor web básico y manejar diferentes tipos de solicitudes HTTP. A través de los objetivos establecidos, hemos logrado:

1. **Configurar un Servidor Básico con Echo:**
   - Iniciamos nuestro servidor Echo y configuramos el puerto de escucha para que la aplicación esté lista para recibir solicitudes entrantes.
2. **Responder con Mensajes desde el Servidor:**
   - Aprendimos a devolver mensajes simples desde el servidor Echo utilizando las funciones `c.String` y `c.JSON`, lo que nos permite proporcionar respuestas de texto plano y JSON a los clientes.
3. **Crear Rutas y Manejar Solicitudes:** - Definimos rutas adicionales en nuestra aplicación Echo para manejar distintos tipos de solicitudes, lo que nos permitió responder con mensajes específicos para cada ruta definida, brindando así una experiencia más dinámica a los usuarios. Hemos adquirido una comprensión sólida de los conceptos fundamentales necesarios para construir servidores web utilizando Echo. Estamos preparados para avanzar y explorar características más avanzadas de Echo, así como para construir aplicaciones web más complejas en el futuro.

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