Lección 5 - Uso de plantillas y carga de contenido estático

Tabla de contenido

Introducción

En esta lección, exploraremos cómo integrar plantillas HTML y servir contenido estático, como archivos CSS y JavaScript, junto con el framework Echo de Go. La combinación de plantillas y contenido estático es esencial para crear páginas web dinámicas y atractivas.

Objetivos

1. **Integrar y Configurar Plantillas HTML en Echo:**
   - Profundizar en la integración de plantillas HTML en proyectos Echo. Veremos cómo configurar Echo para utilizar un motor de renderizado de plantillas y cómo organizar los archivos de plantillas en nuestra aplicación.
2. **Cargar y Servir Contenido Estático:**
   - Aprender a cargar y servir contenido estático, como archivos CSS, imágenes y scripts JavaScript, utilizando Echo. Esto mejorará la apariencia y la interactividad de nuestra aplicación.
3. **Pasar Datos a las Plantillas y Realizar Renderizado Dinámico:**
   - Comprender cómo pasar datos desde los controladores a las plantillas y realizar un renderizado dinámico. Esto nos permitirá mostrar información dinámica en las páginas web generadas.

Desarroll

Integrar y Configurar Plantillas HTML en Echo

Paso 1: Configurar el Proyecto

- Crearemos un directorio `templates` en la raíz del proyecto para almacenar nuestros archivos de plantillas HTML.

```
# Crear directorio
$ mkdir templates

# Crear archivo HTML index.html
$ touch templates/index.html
```

También, crearemos el directorio `static` y dentro de él los subdirectorios `css` y `js`, con sus respectivos archivos de estilo y script.

```
$ mkdir static
$ mkdir static/css static/js
$ touch static/css/styles.css
$ touch static/js/script.js
```

En este ejemplo, hemos creado un directorio llamado `templates` para almacenar nuestras plantillas HTML y un directorio llamado `static` para almacenar nuestro contenido estático, como archivos CSS y JavaScript.

Paso 2: Crear una Plantilla HTML Dentro del directorio `templates`, crearemos el archivo `index.html`. Este será nuestro archivo de plantilla principal donde renderizaremos nuestro contenido dinámico.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{.Title}}</title>
    <link rel="stylesheet" href="/static/css/styles.css">
</head>
<body>
    <h1>{{.Heading}}</h1>
    <p>{{.Message}}</p>
    <script src="/static/js/script.js"></script>
```

```
</body>
</html>
```

En el archivo `routes.go`, configuraremos las rutas y la carga de contenido estático.

```
package routes

import (
    "html/template"
    "io"
    "net/http"
    "github.com/labstack/echo/v4"
)

// TemplateRenderer es una estructura que se utiliza para renderizar
plantillas HTML
type TemplateRenderer struct {
    templates *template.Template
}

// Render es un método de TemplateRenderer que renderiza una plantilla
HTML
func (t *TemplateRenderer) Render(w io.Writer, name string, data
interface{}, c echo.Context) error {
    return t.templates.ExecuteTemplate(w, name, data)
}

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(e *echo.Echo) {
    // Crear una instancia de TemplateRenderer
    renderer := &TemplateRenderer{
        templates: template.Must(template.ParseGlob("templates/*.html")),
    }

    // Configurar el renderizador de plantillas en Echo
    e.Renderer = renderer

    // Configurar la ruta para cargar la página principal
    e.GET("/", func(c echo.Context) error {
        // Renderizar la plantilla index.html con los datos proporcionados
        return c.Render(http.StatusOK, "index.html",
map[string]interface{}{
            "Title":   "Mi Aplicación",
```

```
            "Heading": "¡Hola, mundo!",
            "Message": "Bienvenido a mi aplicación web con Echo y
plantillas HTML.",
        })
    })

    // Configurar Echo para servir contenido estático
    e.Static("/static", "static")
}
```

Cargar y Servir Contenido Estático

Actualizar el contenido del archivo `.css`:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(to bottom, #e2cd64, #f1a044);
}

h1 {
    color: #3a3a3a;
    margin-bottom: 20px;
}

p {
    color: #4e4e4e;
    font-size: 18px;
    line-height: 1.5;
    margin-bottom: 20px;
}
```

Actualizar el contenido del archivo `.js`:

```
function changeMessage() {
    const messageElement = document.getElementById("message");
```

```
    const currentMessage = messageElement.textContent;

    // Lista de mensajes alternativos
    const alternateMessages = [
        "¡Hola de nuevo!",
        "¡Bienvenido de vuelta!",
        "¡Qué tengas un buen día!"
    ];

    // Generar un mensaje aleatorio que no sea igual al mensaje actual
    let newMessage;
    do {
        newMessage = alternateMessages[Math.floor(Math.random() *
alternateMessages.length)];
    } while (newMessage === currentMessage);

    // Actualizar el contenido del elemento de mensaje
    messageElement.textContent = newMessage;
}
```

El body del HTML para que funcione el botón, tiene que quedar de la siguiente manera:

```
    <body>
        <h1>{{.Heading}}</h1>
        <p id="message">{{.Message}}</p>
        <button onclick="javascript: changeMessage();">Actualizar
mensaje</button>
        <script src="/static/js/script.js"></script>
    </body>
```

Hacer las pruebas en el navegador.

Subir los cambios a GitHub:

```
$ git add .
$ git commit -m "Uso de plantillas y carga de contenido estático con Echo"
$ git push
```

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

Conclusión

Hemos comprendido la importancia de las plantillas HTML y el contenido estático, así como también hemos explorado cómo cargarlos y utilizarlos en nuestra aplicación web. Con este conocimiento, estás preparado para comenzar a desarrollar aplicaciones web más complejas utilizando Echo y aprovechar al máximo sus características y funcionalidades.

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