

# Lección 6 - Uso de plantillas y carga de contenido estático

## Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
  - [Integrar y Configurar Plantillas HTML en Gin](#)
    - [Configurar el proyecto](#)
    - [Crear una plantilla HTML](#)
    - [Cargar y Servir Contenido Estático](#)
- [Conclusión](#)
- [Conoce más del autor](#)

## Introducción

En esta clase, exploraremos cómo utilizar plantillas HTML y servir contenido estático, como archivos CSS, imágenes y archivos JavaScript, junto con el Framework Gin de Go. La combinación de plantillas y contenido estático es fundamental para crear páginas web dinámicas y atractivas.

---

## Objetivos

### 1. Integrar y Configurar Plantillas HTML en Gin:

- Profundizar en la integración de plantillas HTML en proyectos Gin. Veremos cómo configurar Gin para utilizar el motor de renderizado de plantillas y cómo organizar los archivos de plantillas.

### 2. Cargar y Servir Contenido Estático:

- Aprender a cargar y servir contenido estático, como archivos CSS, imágenes y scripts JavaScript, utilizando Gin. Esto mejorará la apariencia y la interactividad de la 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 permitirá mostrar información dinámica en las páginas web generadas.
-

# Desarrollo

## Integrar y Configurar Plantillas HTML en Gin

### Configurar el proyecto

- Debemos crear un directorio `templates` en la raíz del proyecto para almacenar sus archivos de plantillas HTML.

```
# Creando directorio
$ mkdir templates

# Creando fichero HTML index.html
$ touch templates/index.html
```

También, necesitamos crear el directorio `static` y los subdirectorios `css` y `js`, con sus respectivos ficheros, tanto de estilo como el 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.

### Crear una plantilla HTML

Dentro del directorio `templates`, el fichero `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 fichero routes.go quedaría de la siguiente manera:

```
package routes

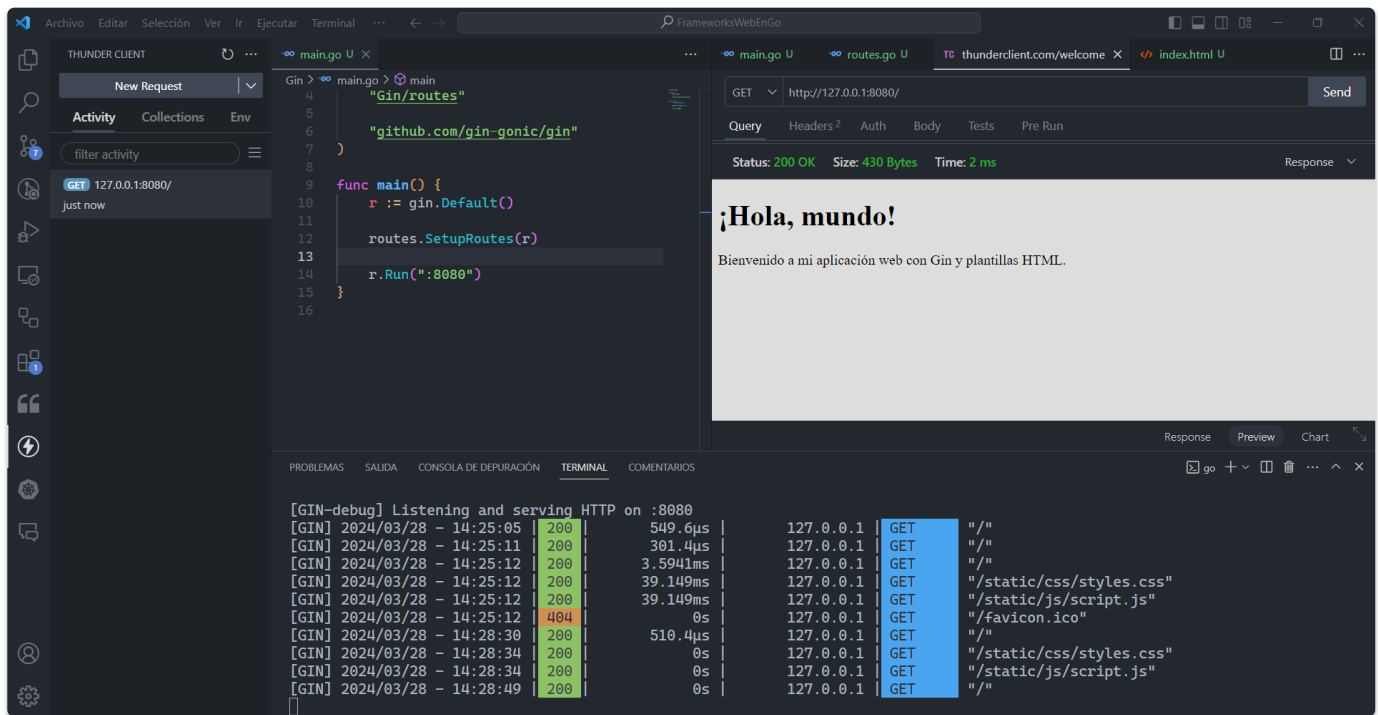
import (
    "net/http"
    "github.com/gin-gonic/gin"
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(r *gin.Engine) {

    // Configurar la ubicación de los archivos de plantillas HTML
    r.LoadHTMLGlob("templates/*")

    // Configurar la ruta para cargar la página principal
    r.GET("/", func(c *gin.Context) {
        // Renderizar la plantilla index.html
        c.HTML(http.StatusOK, "index.html", gin.H{
            "Title":    "Mi Aplicación",
            "Heading":  "¡Hola, mundo!",
            "Message":  "Bienvenido a mi aplicación web con Gin y
plantillas HTML.",
        })
    })

    // Configurar Gin para servir contenido estático
    r.Static("/static", "./static")
}
```

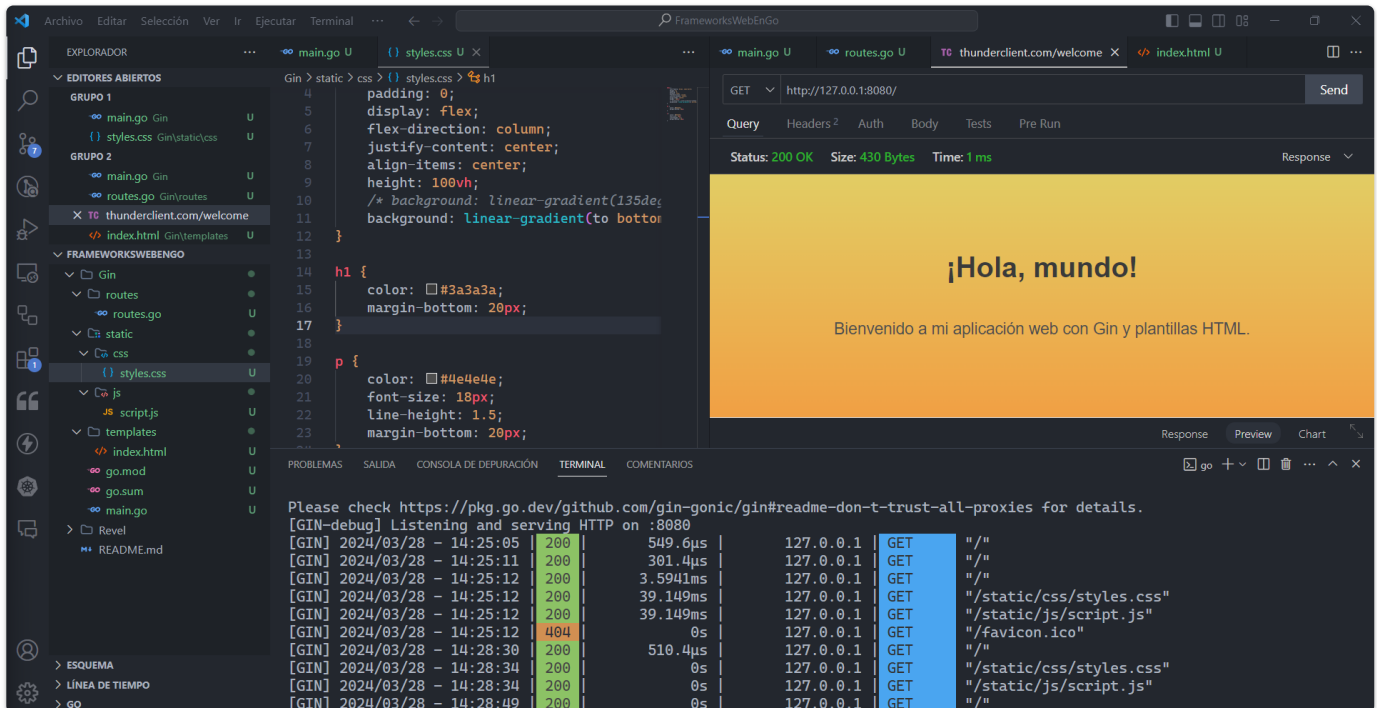


## Cargar y Servir Contenido Estático

Actualizar el contenido .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(135deg, #e0c3fc, #8ec5fc); */  
  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 .js:

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

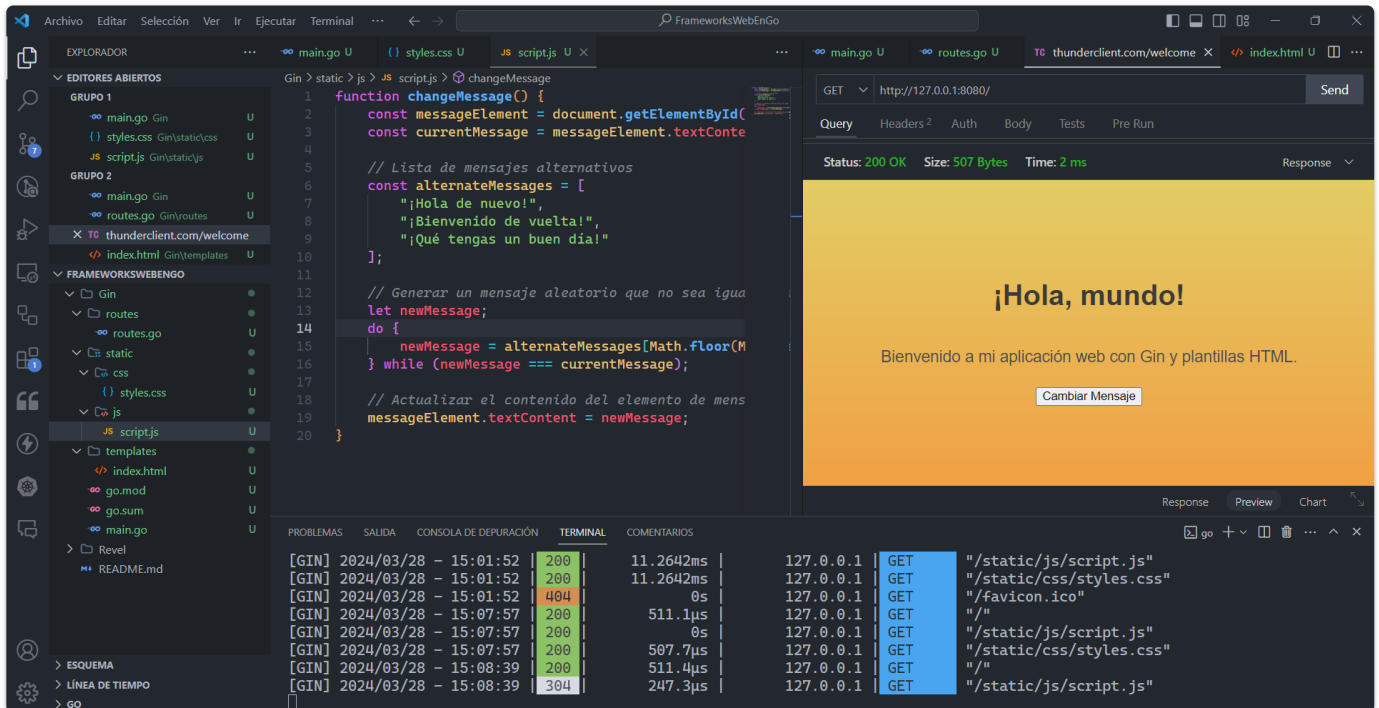
    // Lista de mensajes alternativos
    const alternateMessages = [
        "¡Hola de nuevo!",
        "¡Bienvenido de vuelta!",
        "¡Qué tengas un buen día!"
    ];

    // 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;
```

```
}
```



## Subir los cambios a GitHub:

```
$ git add .
$ git commit -m "Uso de plantillas y carga de contenido estático"
$ 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 Gin y aprovechar al máximo sus características y funcionalidades.

Con estos pasos, están listos para utilizar plantillas y cargar contenido estático en su aplicación web Gin.

## Conoce más del autor

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

 <b>Red Social</b>	 <b>Enlace</b>
 Página web	<a href="https://jersonmartinez.com">jersonmartinez.com</a>
 LinkedIn	<a href="#">Jerson Martínez - DevOps Engineer</a>
 Canales de YouTube	<a href="#">DevOpsea</a>   <a href="#">Side Master</a>
 GitHub	<a href="#">Perfil en GitHub</a>
 Twitter (X)	<a href="#">@antoniomorenosm</a>