Guía de ejercicios - Introducción a Node (II)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada: Menú compartido

Desarrollar un servidor con Express que disponibilice 3 rutas para renderizar 3 vistas diferentes: Inicio, Galería y Contactos. El objetivo será utilizar los parciales de handlebars para que las 3 vistas compartan el mismo menú de navegación.

Consideraciones previas

Necesitamos realizar los siguientes puntos antes de comenzar con el ejercicio:

1. Crear una carpeta “componentes” dentro de la carpeta “views” y crea un archivo llamado “Menu.handlebars” con el siguiente código:

```
<ul style="background: black; padding: 5px">
    <li><a href="/"> Inicio </a></li>
    <li><a href="/Galeria"> Galeria </a></li>
    <li><a href="/Contactos"> Contactos </a></li>
    <li>Número: {{numero}} </a></li>
</ul>

<style>
    li { display: inline-block }
    a, li { color: white }
</style>
```

Como puedes ver, queremos en el último ítem de la lista imprimir el valor de un parámetro, eso se parece bastante a la importación de un componente, pero la diferencia está en que en la renderización de variables no se usa el símbolo “mayor” (>).

2. Crear un archivo llamado “Inicio.handlebars” dentro de la carpeta “views” con el siguiente código:

```
{{> Menu numero="1"}}
<h1>Esta es la página de Inicio</h1>
```

3. Crear un archivo llamado “Galeria.handlebars” dentro de la carpeta “views” con el siguiente código:

```
{{> Menu numero="2" }}
```

```
<h1>Esta es la página de Galería</h1>
```

4. Crear un archivo llamado “Contactos.handlebars” dentro de la carpeta “views” con el siguiente código:

```
{{> Menu numero="3"}}
<h1>Esta es la página de Contactos</h1>
```

Como puedes notar, estamos pasándole al parcial “Menu” un parámetro “numero” en cada template con números del 1 al 3 respectivamente, esto lo hacemos para comprobar el paso de parámetros a un parcial o componente. Ahora que tenemos nuestras vistas y componente creado, sigue los siguientes pasos para la creación de nuestro servidor:

- ***Paso 1:*** Crear un servidor con Express que integre handlebars como motor de plantillas.

- ***Paso 2:*** Usar el método “engine” para definir el objeto de configuración de handlebars, especificando que la ruta para las vistas será **/views** y la ruta en donde se encontrarán los parciales o componentes será **/views/componentes.**

- ***Paso 3:*** Crear una ruta raíz **GET /** que utilice el método render, recibiendo como primer parámetro “Inicio” y como segundo un objeto, que contenga la propiedad “layout” y cuyo valor será “Inicio”. Este objeto será el que defina que se utilizará el archivo **Inicio.handlebars** para renderizar esta vista.

- ***Paso 4:*** Crea una ruta raíz **GET /Contactos** que utilice el método render, recibiendo como primer parámetro “Contactos” y como segundo un objeto, que contenga la propiedad “layout” y cuyo valor será “Contactos”.

- ***Paso 5:*** Crea una ruta raíz **GET /Galeria** que utilice el método render, recibiendo como primer parámetro “Galeria” y como segundo un objeto, que contenga la propiedad “layout” y cuyo valor será “Galeria”.

```
// Paso 1
const express = require("express");
const app = express();
const exphbs = require("express-handlebars");

app.listen(3000, () => {
  console.log("El servidor está inicializado en el puerto 3000");
});
```

```
app.set("view engine", "handlebars");

// Paso 2
app.engine(
  "handlebars",
  exphbs({
    layoutsDir: __dirname + "/views",
    partialsDir: __dirname + "/views/componentes/",
  })
);

// Paso 3
app.get("/", function (req, res) {
  res.render("Inicio", { layout: "Inicio" });
});

// Paso 4
app.get("/Contactos", function (req, res) {
  res.render("Contactos", { layout: "Contactos" });
});

// Paso 5
app.get("/Galeria", function (req, res) {
  res.render("Galeria", { layout: "Galeria" });
});
```

Probemos esto, levanta el servidor e ingresa a las siguientes direcciones una por una:

- http://localhost:3000/
- http://localhost:3000/Galeria
- http://localhost:3000/Contactos

Deberás ver lo que te muestro en las siguientes imágenes:

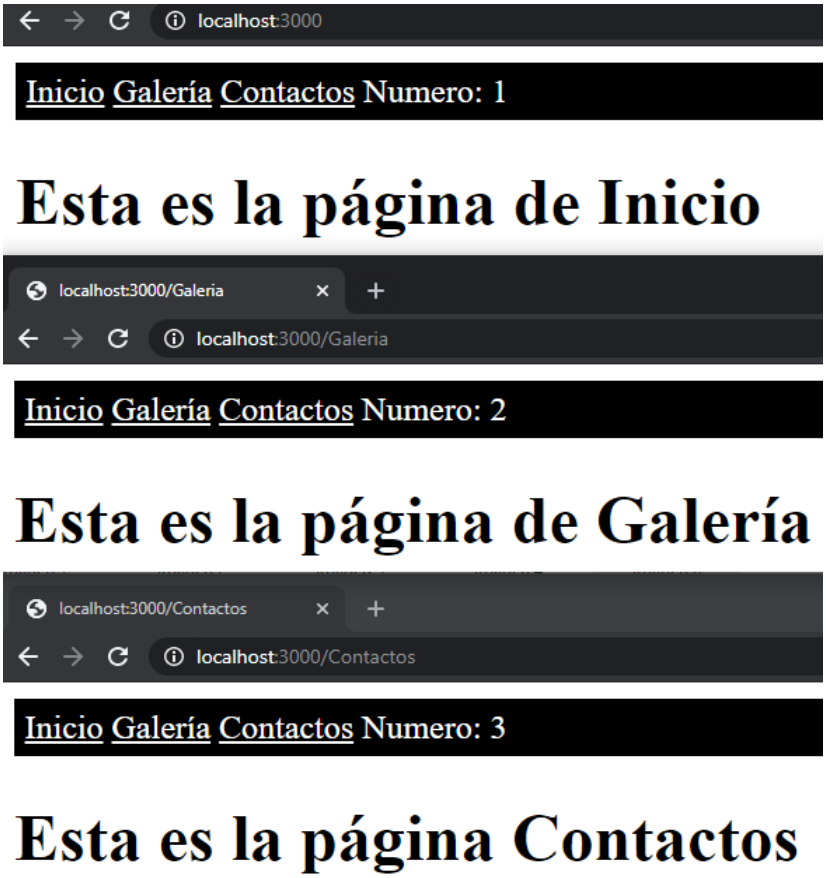

Imagen 1. Renderización de vistas que comparten el mismo parcial.
Fuente: Desafío Latam

Como puedes ver, se está renderizando el componente "Menu" en las 3 vistas y de esta manera hemos evitado tener que copiar y pegar el mismo código, además, hemos comprobado el paso de parámetro al parcial con valores diferentes en cada vista.

Actividad guiada - Ver mis ventas

Agregar a nuestro servidor una ruta **GET /ventas** que use los helpers each, if y unless para mostrar las ventas de un usuario, enviadas como parámetro en el método render.
Sigue los siguientes pasos para resolver este ejercicio:

- ***Paso 1:*** Crear una ruta **GET /ventas** en el servidor.
- ***Paso 2:*** Utilizar el método render para renderizar una vista "Ventas" que pase como parámetro las propiedades "usuario" con el nombre de usuario y un arreglo de números llamado "ventas".

```
// Paso 1
app.get("/ventas", function (req, res) {
  // Paso 2
  res.render("Ventas", {
    layout: "Ventas",
    usuario: "Maria José",
    ventas: [14990, 42490, 22500],
  });
});
```

- ***Paso 3***: Crear un archivo en la carpeta "views" que se llame "Ventas.handlebars" y que contenga el helper "if" para validar que existe valor en el parámetro usuario.

- ***Paso 4:*** Utilizar el helper "each" para iterar el arreglo "ventas" e imprimir sus elementos dentro de una lista desordenada.

- ***Paso 5:*** Utilizar el helper "unless" para imprimir un mensaje que indique que se debe iniciar sesión en caso de que el parámetro "usuario" sea "false".

```
{{!-- Paso 3 --}}
{{#if usuario}}
<h1>
    Bienvenid@: {{usuario}}
</h1>
<h3>Tus ventas hoy fueron las siguientes:</h3>
<ul>
    {{!-- Paso 4 --}}
    {{#each ventas}}
    <li>${{this}}</li>
    {{/each}}
</ul>
{{/if}}

{{!-- Paso 5--}}
{{#unless usuario}}

<h1>Inicie sesión para ver sus ventas</h1>

{{/unless}}
```

Ahora levanta el servidor y en caso de darle valor a la propiedad "usuario" obtendrás algo similar a la siguiente imagen:

Bienvenid@: Maria José

Tus ventas hoy fueron las siguietes:

- $14990
- $42490
- $22500

Imagen 2. Renderización de una vista, caso de usuario con valor.
Fuente: Desafío Latam

Como puedes ver, hemos recibido la renderización del bloque if y dentro la iteración del arreglo “ventas” gracias al helper “each”.

¿Qué sucedió con el bloque unless? Recordemos que este bloque mostrará contenido si el parámetro que definimos es “false”. Para comprobarlo cambia el valor de la propiedad “usuario” dentro del método “render” por “false” y si vuelves a consultar la ruta **/ventas** verás lo que te muestro en la siguiente imagen:

Inicie sesión para ver sus ventas

Imagen 3. Renderización de una vista, caso de usuario sin valor o con valor false.
Fuente: Desafío Latam

Excelente, ahora el contenido que mostramos es dinámico y dependerá del valor de la propiedad usuario y del arreglo ventas. Considera que esto es muy escalable y podemos mezclarlo con la obtención de parámetros o payload enviados desde una aplicación cliente, y de esta manera construir sistemas cada vez más complejos y dinámicos.

¡Manos a la obra! - Saludo.txt

Desarrollar una aplicación en Node que cree un archivo llamado “Saludo.txt” y tenga como contenido el mensaje “Hola Mundo!”. Justo luego de ser creado este archivo con el método writeFile, imprime el contenido de este mismo dentro de su callback con el método readFile.

¡Manos a la obra! - Registro de gastos

Basado en el ejercicio de los formularios HTML y el servidor para la creación de Repertorio.txt, crea otro sistema tipo CRUD que sirva como registros de gastos, es decir, en este ejercicio el planteamiento del problema será un control financiero personal en donde la persona podrá registrar los gastos que tuvo esta semana, además de consultarlo, poder renombrar el archivo de texto llamado Gastos.txt y eliminarlo en su totalidad.

En el lado del servidor deberás crear las 4 rutas con su lógica correspondiente con el módulo file System.

Puedes ocupar los mismos formularios HTML del ejercicio de Repertorio.

¡Continúa aprendiendo y practicando!

Soluciones

1. Desarrollar una aplicación en Node que cree un archivo llamado “Saludo.txt” y tenga como contenido el mensaje “Hola Mundo!”. Justo luego de ser creado este archivo con el método WriteFile, imprime el contenido de este mismo dentro de su callback con el método readFile.

```
const fs = require('fs')
fs.writeFile('Saludo.txt', 'Hola Mundo!', 'utf8', () => {
   fs.readFile('Saludo.txt', 'utf8', (err, data) => {
```

```
    console.log(data)
  })
})
```

2. Basado en el ejercicio de los formularios HTML y el servidor para la creación de Repertorio.txt, agrega 2 formularios para el renombramiento del archivo y para la eliminación del mismo.

   En el lado del servidor deberás crear las rutas y la lógica correspondiente con el módulo File system.

```
<!-- Primer Formulario -->
<form action="/crear">
  Nombre del archivo:
  <input name="nombre" />
  <br />
  Contenido:
  <textarea name="contenido"></textarea>
  <br />
  <button>Crear</button>
</form>
<hr />
<!-- Segundo Formulario -->
<form action="/leer">
  Nombre del archivo:
  <input name="nombre" />
  <br />
  <button>Leer</button>
</form>

<!-- Tercer Formulario -->
<form action="/renombrar">
  Nombre del archivo:
  <input name="nombre" />
  <br />
  <button>Renombrar</button>
</form>

<!-- Cuarto Formulario -->
<form action="/eliminar">
  Nombre del archivo:
  <input name="nombre" />
  <br />
  <button>eliminar</button>
</form>
```

```
const express = require('express')
const app = express()
const fs = require('fs').promises

app.listen(3000, console.log("SERVER ON"))

app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html")
})

app.get("/crear", async (req, res) => {
  const { nombre, contenido } = req.query
  try {
    await fs.writeFile(nombre, contenido)
    res.send("Archivo creado con éxito!")
  } catch (error) {
    res.status(500).send("Algo salió mal...")
  }
})

app.get("/leer", async (req, res) => {
  const { nombre } = req.query
  try {
    const data = await fs.readFile(nombre)
    res.send(data)
  } catch (error) {
    res.status(500).send("Algo salió mal...")
  }
})

app.get("/renombrar", async (req, res) => {
  const { nombre } = req.query
  try {
    await fs.rename('Gastos.txt', nombre)
    res.send(`Archivo Gastos.txt renombrado por ${nombre}`)
  } catch (error) {
    res.status(500).send("Algo salió mal...")
  }
})

app.get("/eliminar", async (req, res) => {
  const { nombre } = req.query
  try {
    await fs.unlink(nombre)
    res.send(`Archivo ${nombre} eliminado con éxito`)
```

```
  } catch (error) {
    res.status(500).send("Algo salió mal...")
  }
})
```