Introducción a Node

Sirviendo contenido web

{desafío} latam_

Implementar un servidor web de contenidos estáticos y dinámicos utilizando motores de plantillas acorde al entorno Node Express para dar solución a un problema.

- Unidad 1:
  Introducción a Node
- Unidad 2:
  Node y el gestor de paquetes
- Unidad 3:
  Persistencia

Te encuentras aquí

{desafío}
latam_

¿Qué aprenderás en esta sesión?

- *Describir las características y el rol de un motor de plantillas para el despliegue dinámico de contenidos.*
- *Utilizar motor de plantillas Handlebars para el despliegue de contenido dinámico.*
- *Implementar una vista dinámica utilizando renderización de objetos, variables, páginas parciales y helpers para resolver un problema.*

{desafío} latam_

¿Cómo se puede leer el contenido de un archivo utilizando el módulo "File System" en Node?

¿Qué se debe hacer para manejar los métodos asíncronos del módulo "File System" en Node como promesas utilizando la nueva API?

{desafío} latam_

/* Motores de plantilla */

Motores de plantillas

- Los motores de plantillas, sirven como una herramienta que podemos integrar a nuestros servidores con Express.
- Ofrecen renderizar y reutilizar código HTML como partes de un rompecabezas, simulando el principio de las funciones en la programación, lo cual, nos permite contar con una estructura que podemos volver a ocupar haciendo uso de su llamado.
- El objetivo principal de los motores de plantillas, es lograr una mejor organización del template en nuestros proyectos, en especial cuando tenemos varias vistas que comparten contenido.

Motores de plantillas

¿Qué hacen?

Algunas de las tareas que realizan los motores de plantillas en JavaScript incluyen:

1. **Interpolación de datos:** Permiten insertar datos dinámicos dentro de una plantilla HTML utilizando marcadores o placeholders. Estos marcadores se reemplazan por los valores reales cuando se renderiza la plantilla.

1. **Estructuras de control:** Los motores de plantillas pueden manejar estructuras de control, como bucles y condicionales, para generar contenido repetitivo o mostrar elementos basados en ciertas condiciones.

Motores de plantillas

¿Qué hacen?

3. **Reutilización de componentes:** Facilitan la creación y reutilización de componentes HTML que contienen lógica y datos asociados. Esto ayuda a mantener un código más modular y facilita el desarrollo.

3. **Escapado de contenido:** Los motores de plantillas se encargan de asegurar que el contenido dinámico insertado en la plantilla no cause problemas de seguridad, evitando la ejecución de código malicioso (ataques de inyección de scripts) y garantizando una correcta visualización de los datos.

Motores de plantillas

¿Qué hacen?

5. **Filtrado y formateo de datos:** Permiten aplicar filtros y formatos a los datos antes de mostrarlos en la plantilla, lo que facilita la presentación de información de manera adecuada.

5. **Manejo de eventos:** En algunos casos, los motores de plantillas también pueden gestionar eventos generados por los elementos generados en la plantilla, simplificando la interacción con el usuario.

/* Handlebars js */

Handlebars js

Es un motor de plantillas destacado por su velocidad y facilidad de uso. Permite la creación y reutilización de parciales, los cuales no son más que porciones o pedazos de HTML que pueden ser importados y reutilizados, además, cuenta con Helpers que nos ayudarán a renderizar variables y condicionales dentro de nuestro código HTML.

Handlebars js

Instalación

En NPM contamos con un paquete que ya viene configurado para ser integrado con Express, incluso su nombre es express-handlers y podemos instalarlo con el siguiente comando:

```
npm install --save express-handlebars
```

Esto descargará e instalará la biblioteca Handlebars.js y sus dependencias en la carpeta node_modules de tu proyecto.

Handlebars js

Integración y configuración

Ahora que tenemos nuestro motor de plantillas instalado, podemos proceder con su integración y configuración en donde especificaremos los directorios donde estarán ubicados nuestros archivos de extensión “handlebars”.

No estaremos creando documentos puros HTML, sino que usaremos archivos cuya extensión se llaman igual que el motor de plantillas, no obstante, debes saber que esto pudiéramos cambiarlo a nuestro gusto y lo revisaremos a continuación.

Handlebars js

Integración y configuración

Antes de iniciar con el desarrollo de nuestro primer ejercicio guiado con motor de plantilla, sigue los siguientes pasos:

1. Crea una carpeta nueva y abrela con tu editor preferido.
2. Instala Express y el express-handlebars.
3. Crea una carpeta llamada “views”. Esta carpeta debe llamarse de esta manera porque por defecto handlebars la buscará con ese nombre.
4. Dentro de la carpeta “views”, crea un archivo llamado main.handlebars.

Con lo anterior listo, ha llegado el momento de crear un servidor con Express e integrar handlebars.

Demostración "Estrenando handlebars"

Ejercicio guiado

Estrenando handlebars

Desarrollar un servidor con Express que utilice handlebars para renderizar una vista con una cabecera HTML que diga “Hola mundo! Usando ahora motores de plantillas =D”. Sigue los siguientes pasos para resolver este ejercicio de forma progresiva:

- **Paso 1:** Crear un servidor con Express e incluir el paquete express-handlebars en tus importaciones iniciales.

```
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");
});
```

{desafío} latam_

Ejercicio guiado

Estrenando handlebars

- **Paso 2:** Ocupar un método de la instancia “app” llamado “set”. Este método lo que hace es definir una configuración para ser utilizada por Express, en nuestro caso la usaremos para integrar handlebars como motor de plantillas de la siguiente manera:

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

Ejercicio guiado
Estrenando handlebars

- **Paso 3:** Configurar el motor de plantilla, para esto debemos usar el método “engine”, el cual define el motor de plantillas que utilizaremos en nuestro servidor con Express. Este método recibe los siguientes parámetros:
    - **Primer parámetro:** Define la extensión de los archivos que handlebars identificará como “vistas” o layouts para nuestra aplicación, en nuestro caso será el mismo nombre del motor, es decir: “handlebars”.
    - **Segundo parámetro:** Recibe la instancia de express-handlebars que importamos al inicio del código. Esta instancia es realmente un método que recibe un objeto de configuración en el que definiremos la propiedad “layoutsDir”, y cuyo valor será la ruta del directorio donde tendremos las vistas o layouts que utilizaremos en nuestra aplicación.

```
app.engine(
  "handlebars",
  exphbs.engine({
    layoutsDir: __dirname + "/views",
  })
);
```

Ejercicio guiado
Estrenando handlebars

- **Paso 4:** Crear la ruta raíz del servidor, como respuesta usaremos el método “render” del objeto response, el cual recibe los siguientes parámetros:
    - **Primer parámetro:** Define el nombre de la vista que queramos renderizar, en nuestro caso es “main”.
    - **Segundo parámetro:** Recibe un objeto con el que podemos especificar el layout que queremos que se renderice, no obstante, por defecto handlebars buscará un archivo llamado “main.handlebars” por lo que no será necesario incorporarlo. Además, no es necesario colocar la extensión pues automáticamente es reconocida gracias a la configuración previa.

```
app.get("/", (req, res) => {
  res.render("main");
});
```

Ejercicio guiado

Estrenando handlebars

Bien, tenemos escrito todo lo necesario a nivel de servidor, pero ¿Qué sucede con el archivo de extensión handlebars? Es momento de darle contenido. Utiliza el siguiente código HTML para este archivo:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Document</title>
</head>
<body>
   <h1>Hola mundo! Usando ahora motores de plantillas =D</h1>
</body>
</html>
```

Ejercicio guiado

Estrenando handlebars

Es momento de probar todo lo que hicimos, así que levanta tu servidor y consulta la dirección http://localhost:3000/, deberás recibir lo que te muestro en la siguiente imagen:

¡Excelente! Con esto has logrado renderizar tu primera plantilla con handlebars desde un servidor con Express.

{desafío} latam_

/* Parciales */

Parciales

- Handlebars al igual que los demás motores de plantillas, permite la reutilización de plantillas a través de partials.

- Los partials son plantillas normales de handlebars, que otras plantillas pueden llamar directamente y así formar un rompecabezas, en donde podemos tener una estructura principal y llamar a las piezas que conformarán la vista.

- De forma predeterminada, los partials son buscados en una carpeta llamada “partials” dentro de la carpeta “view”, no obstante, se puede configurar esta ruta y definir el nombre que quieras para la carpeta que contendrán las piezas de nuestra aplicación

Parciales

Importando parciales

- Los parciales son estructuras o porciones de código que podemos reutilizar a través de su llamado

- La forma de invocar un partial dentro de una vista es con las dobles llaves ({{ }}) y con el signo de “mayor” (>).

```
{{> Menu }}
```

Parciales

Envío de parámetros a los partials

- Se ha mencionado que el concepto de “parciales” es comparable al concepto de “funciones”, por el hecho de ser estructuras reutilizables, pero incluso además de eso, también comparten la cualidad de poder recibir parámetros

- La forma de enviar variables o “argumentos” a un parcial, es la siguiente:

```
{{> Cards
     nombre="Inteligencia Artificial"
     imagen="img/inteligencia-artificial.jpg"
     descripcion="Es la inteligencia llevada a cabo por máquinas."
 }}
```

Parciales

Envío de parámetros a los partials

El resultado del código anterior sería como lo que se muestra en la siguiente imagen:

- La información dentro de la tarjeta es la que se está definiendo como parámetros del parcial.

/* Helpers */

Helpers

¿Qué son los helpers?

Los Helpers son funciones ofrecidas por handlebars para la reproducción de datos que recibimos en los parciales, bien sea por parámetro definido en su llamado desde otro parcial o por definición desde el render. La forma de importarlos es muy parecida a la de un parcial, pero la diferencia es el símbolo, en este caso es un hash o también conocido como “gato” (#).

Entre los helpers de este motor de plantilla tenemos:

- if
- Else
- Unless
- Each

Helpers

If

- **if**: Funciona únicamente con la condición booleana, es decir, evalúa si el dato es true o false y su sintaxis es la siguiente:

```
{{#if nombre }}
<h3>Bienvenido: {{ nombre }} </h3>
{{/if }}
```

Nota: Como puedes notar, no se usan paréntesis para definir la condición ni llaves para delimitar el bloque de renderización, sino que se habilita el inicio del helper, seguido de las instrucciones que queremos mostrar y para cerrar el bloque correspondiente al “if”, se ocupa la misma sintaxis pero con el slash ‘’/” seguido del nombre del helper.

Helpers
Else

- **Else:** Debe existir siempre dentro del bloque de renderización del **if** y su sintaxis es la siguiente:

```
{{#if nombre }}
<h3>Bienvenido: {{ nombre }} </h3>

{{else}}

<h3>Aún no ha ingresado ningún usuario</h3>

{{/if }}
```

Helpers

Unless

- **Unless:** Parecido al if, este helper sólo muestra contenido si el dato en evaluación es “false”, por ejemplo, cuando se intenta ingresar a un sistema y el usuario enviado a la validación no se encuentra. Por ejemplo:

```
app.get('/', function (req, res) {
 res.render('Home', {
   isLogged: false
 })
})
```

Nota: Como puedes ver, es igual a la definición que usamos para especificar el layout de una vista, sin embargo, la propiedad “layout” se puede considerar como una palabra reservada de handlebars, por lo que definir una propiedad con cualquier otro nombre será interpretado por el mismo motor de plantilla como un parámetro para ese parcial.

Helpers

Unless

- La renderización dentro del parcial para el “unless” sería como te muestro a continuación:

```
{{#unless isLogged}}
 <p>El usuario no está registrado</p>
{{/unless}}
```

Helpers

Each

- Each: Es el iterador de arreglos que podemos usar dentro de las plantillas, su sintaxis incluye la palabra reservada “this” para indicar cada elemento del arreglo, muy parecido al iterador each de jQuery o al forEach de vanilla js.

Por ejemplo, se itera un arreglo que contiene, como elementos, cadenas de texto con el nombre de los 3 lenguajes principales del front end. Se envía el parametro desde el método render dentro de la ruta en el servidor

```
app.get("/", function (req, res) {
  res.render("Inicio", {
    layout: "Inicio",
    lenguajesFrontend: ["HTML", "CSS", "Javascript"],
  });
});
```

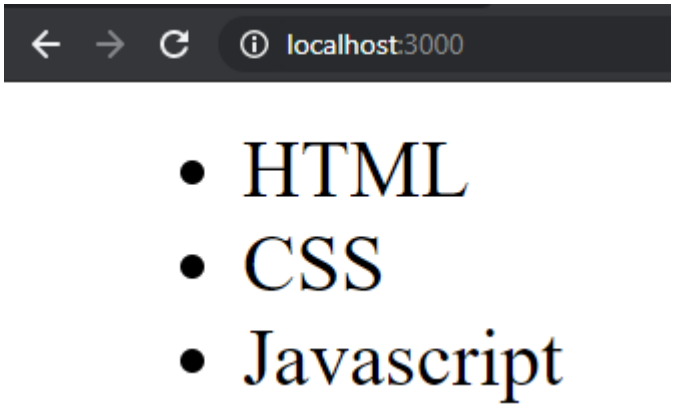
Helpers
Each

Ahora, el código dentro de la vista sería:

```
<ul>
   {{#each lenguajesFrontend}}

   <li>{{this}}</li>

   {{/each}}
</ul>
```



Se está integrando la etiqueta <li> de una lista desordenada dentro del bloque “each” de handlebars, así se logra que este elemento se repita por cada iteración del arreglo lenguajesFrontend e imprima el valor del elemento con el “this”.

¿Cuál es la principal ventaja de utilizar motores de plantillas como Handlebars en el desarrollo de aplicaciones web con Node.js y Express?

{desafío} latam_

Próxima sesión...

- *Devolviendo un sitio web de contenido dinámico*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales