OBJETIVO

En esta guía aprenderemos a usar los motores de plantilla y modularizar rutas, para que a futuro puedas implementarlo en tus aplicaciones.

Motores de plantilla

Un motor de plantilla permite crear variables en un documento html, los cuales serán construidos al momento de generarlos. Conjuntamente con express se puede adicionar un motor de plantillas. En Nodejs existen diferentes motores, los más populares son: Pug, Jade, handlebarjs, mustachejs,EJS, etc. Puede consultar la lista completa de motores de plantilla compatibles con express. En esta guía usaremos el motor de plantillas handlebars.

Handlebars (hbs)

Handlebars es un motor de plantilla simple pero poderoso, que necesita un solo objeto de entrada para generar HTML u otros formatos de texto.

Para poder usar Handlebars o cualquier otro motor de plantillas en nuestro proyecto, debemos instalar adicionalmente la dependencia `consolidate` el cual es una libreria wrapper de diversos motores de plantillas.

```
$ npm install consolidate handlebars
```

Al inicio de la definición de las rutas debemos inicializar nuestro motor de plantillas de la siguiente forma:

```
const express = require("express");
const app = express();
const engine = require('consolidate');
app.engine("hbs",engine.handlebars); //.hbs extension
app.set("views","./views");
app.set("view engine","hbs");

app.get('/',function(req,res){
    res.send('<h1>Hola</h1>');
});

app.listen(3000,function(){
    console.log("servidor ejecutando");
});
```

El código anterior indica que usaremos el motor de plantillas handlebars (`hbs`) y nuestros archivos de plantillas (.hbs) estarán ubicados en la carpeta `./views`

Para añadir nuestra primera plantilla creamos un archivo `template1.hbs` dentro de la carpeta views, Creamos una variable `nombre` usando llaves dobles `{{nombre}}`

plantilla1.hbs

```
<h1>
Bienvenido,  {{nombre}}
</h1>
```

Invocamos a la plantilla usando el método `render` y como segundo parámetro enviamos los datos de nuestra plantilla, de la siguiente forma:

```
app.get('/datos',function(req,res){
    res.render('plantilla1',{nombre:'Juan Gonzales'});
});
```

Comentarios

Los comentarios en las plantillas hbs, se escribe de la siguiente forma:

```
{{!Esto es un comentario que estará visible solamente en la plantilla hbs}}
```

```
<!-- Este comentario estará visible en el html generado -->
```

Bloques

Los bloques son expresiones que se abren con `{{#}}` y se cierran con `{{/}}`

Helpers

Mediante el uso de helpers podemos ejecutar lógicas simples usando funcionalidades auxiliares. Un helper puede ser invocado de la siguiente forma:

```
{{#nombreDelHelper parametro1,parametro2 ...}}
El contenido va aqui
{{/nombredelHelper}}
```

Algunos de los helpers que vienen integrados son: `#if` `#each` `#unless` `#with`. La lista completa lo puedes consultar desde el siguiente enlace: https://handlebarsjs.com/guide/block-helpers.html

helper #if

El helper **#if** se en hbs se escribe de la siguiente forma:

```
{{#if countries}}
```

```
  The countries are present.
{{else}}
  The countries are not present.
{{/if}}
```

Si queremos verificar algo que sea falo podemos invertir **#if** por **#unless**:

```
<div class="entry">
{{#unless license}}
<h3 class="warning">WARNING: This entry does not have a license!</h3>
{{/unless}}
</div>
```

helper #with

El helper #with permite acceder a la propiedad de un objeto de forma directa. Ejemplo

Entrada:

```
{
  person: {
    firstname: "Bill",
    lastname: "Gates",
  },
}
```

plantilla:

```
 {{#with person}}
    {{firstname}} {{lastname}}
 {{/with}}
```

Salida:

```
 Bill Gates
```

bucle #each

El bucle #each permite recorrer una lista: Entrada:

```
{
  people: [
    "Yehuda Katz",
    "Alan Johnson",
    "Charles Jolley",
  ],
}
```

Plantilla:

```
<ul class="people_list">
  {{#each people}}
    <li>{{this}}</li>
  {{/each}}
</ul>
```

Salida:

```
<ul class="people_list">
    <li>Yehuda Katz</li>
    <li>Alan Johnson</li>
    <li>Charles Jolley</li>
</ul>
```

Opcionalmente se puede añadir la sentencia {{else}} al bloque #each, el cual se ejecutará cuando la lista esté vacia:

```
<ul class="people_list">
  {{#each people}}
    <li>{{this}}</li>
    {{else}}
     <li class="empty">La lista está vacia</li>
  {{/each}}
</ul>
```

Salida en Log

hbs permite mostrar mensajes de log usando la sentencia {{log

```
{{log 'Esta salida se mostrará en el log'}}
```

Se puede pasar cualquier numero de argumentos:

```
{{log 'firstname' firstname 'lastname' lastname}}
```

Plantillas parciales

Una aplicación puede contener diferentes secciones. Con hbs podemos separarlo en diferentes archivos de plantilla cada sección.

```
.
├── app.js
└── views
    ├── home.hbs
    ├── _sidebar.hbs
```

```
    ├── _header.hbs
    ├── home.hbs
    ├── layouts
    |   └── main.hbs
    └── partials
        └── panel1.hbs
        └── panel2.hbs
```

Luego unimos en un archivo principal `home.hbs`

```
{{>header}}
<hr/>
{{>sidebar personas=personas}}
```

```
app.get('/home',function(req,res){
    var partials = {header: '_header', sidebar: '_sidebar'};

    personas = [
        "Yehuda Katz",
        "Alan Johnson",
        "Charles Jolley",
      ]

    res.render('home',{partials:partials,personas:personas});
})
```

Al cargar http://localhost:3000/home notaremos que todos los archivos parciales se fusionaron

Repositorio

El repositorio de este laboratorio se encuentra en la siguiente dirección:
https://github.com/F1852D160/02-plantillasexpress