Coding Bootcamp

Sprint 2

Temario

Temario

- Motores de Templates.
- EJS.
  - Instalación.
  - Uso.
  - Sintaxis.
  - Include.

Motores de Templates

Motores de Templates

Anteriormente vimos que podíamos enviar un archivo HTML como respuesta a una llamada HTTP.

En Express vimos que podemos hacerlo utilizando el método `res.sendFile` y pasándole un archivo HTML (ej: `home.html`).

Sin embargo, los archivos HTML son estáticos. Es decir, tendríamos que tener un archivo HTML distinto para cada página o para cada tipo de respuesta que quisiéramos enviar. Además, tendríamos que repetir código en cada página (ej: para el cabezal o para el pie de página). Esto es muy poco conveniente.

Por suerte, en casi todos los lenguajes de programación existen Motores de Templates o *Template Engines* que facilitan mucho esta tarea.

EJS

EJS

- Es un motor de templates para JavaScript. Es independiente de Node.js y de Express.
- Permite generar archivos HTML de forma dinámica, a partir de código JavaScript (aunque con una sintaxis particular), y aprovechando todas las ventajas de un lenguaje de programación.
- En general vamos a trabajar con archivos con extensión `.ejs` y los colocaremos dentro de una carpeta llamada `views`.
- Documentación: https://ejs.co.
- Otras alternativas pueden ser Pug (antes llamado Jade), Mustache o Edge.

EJS – Instalación

Para instalar EJS en un proyecto de Node.js se debe correr el comando:

```
npm i ejs
```

Si estamos usando Express, es necesario indicarle a nuestra `app` que vamos a usar EJS. Esto lo hacemos en `index.js` (o el archivo principal que estemos usando):

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

EJS – Uso (1/2)

Ejemplo de uso:

Ej: `index.js`

```
app.get("/", function (req, res) {
    res.render("home", { title: "Hack Academy" });
});
```

El método `render` permite llamar a un archivo (`home.ejs`) que está dentro de la carpeta `views` del proyecto. Notar que no es necesario especificar la extensión del archivo ya que anteriormente se le dijo a Express que se usará EJS como motor de templates.

Como segundo parámetro se pasa un objeto con datos que se le quieren pasar a la vista `home`.

EJS – Uso (2/2)

Veamos ahora el archivo `home.ejs`. Su contenido es muy similar al que podría contener un archivo HTML, pero con algunas características especiales:

Ej: `home.ejs`

```
...
    <body>
        <h1><%= title %></h1>
    </body>
</html>
```

La sintaxis `<%= … %>` es propia de EJS. En tiempo de ejecución, EJS sustituye esa etiqueta por el contenido de la variable `title`.

Por mayor detalles entrar a https://ejs.co/#docs.

EJS – Include

EJS – Include (1/3)

Con EJS es posible que un archivo `.ejs` incluya a otro archivo `.ejs`.

Por ejemplo, el archivo `views/home.ejs` podría incluir el archivo `views/partials/header.ejs`:

Ej: `home.ejs`

```
...
    <body>
        <%- include("partials/header") %>
    </body>
</html>
```

Notar el guión luego del primer símbolo de porcentaje. Esto no es un error, así es como se utiliza el `include`.

El uso de la carpeta `partials` no es obligatorio, pero suele ser una buena práctica.

Esto es útil para no repetir código, ya que el cabezal del sitio lo podemos escribir una sóla vez y luego incluirlo desde varias otras páginas.

EJS – Include (2/3)

En caso de necesitar, se le pueden pasar datos al `include`, tal como se le pasan datos a cualquier otra vista (cuando usamos el método `render`).

Ejemplo:

Ej: `home.ejs`

```
...
    <body>
        <%- include("partials/header", { title: "Hack Academy" }) %>
    </body>
</html>
```

⚠️ Notar que cuando se le pasan datos a una vista, esos datos sólo quedan disponibles en esa vista. No quedan disponibles (de forma automática) en las vistas que se están incluyendo. Por ejemplo, si se le pasaron datos a `home.ejs` y se quiere pasarlos también a `header.ejs`, hay que hacerlo explícitamente.

EJS – Include (3/3)

La solución anterior se podría resumir en el siguiente diagrama.