HANDLEBARS (PRÁCTICA)

Tiempo estimado: 25min

El objeto de esta práctica es asentar y consolidar los conceptos presentados en la parte teórica de la lección.

Al finalizarla, el estudiante:

- Habrá configurado Handlebars como motor de plantillas de la aplicación.
- Habrá creado plantillas.
- Habrá definido y configurado la plantilla de diseño.
- Habrá usado bloques.
- Habrá definidos y usado plantillas parciales.
- Habrá definido y usado funciones asistentes.

OBJETIVOS

El objetivo de la práctica es crear una aplicación Express que use Handlebars como motor de plantillas. Esta aplicación debe utilizar una plantilla de diseño, disponer de plantillas, parciales y funciones asistentes.

CREACIÓN DEL PROYECTO

Para comenzar, vamos a crear el proyecto de la aplicación.

1. Abrir una consola.
2. Crear el directorio de la práctica.
3. Ir al directorio de la práctica.
4. Crear el archivo package.json con el siguiente contenido:

```
{
  "name": "express-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "start": "node ./app.js",
    "start-dev": "./node_modules/.bin/nodemon -e hbs,js,json ./app.js"
  },
  "dependencies": {
    "express": "*"
  },
  "devDependencies": {
    "nodemon": "*"
  }
}
```

   Observe el uso de la opción -e de nodemon para indicar las extensiones que deseamos monitorizar del directorio de la aplicación.

5. Crear el archivo index.html:

```
<!doctype html>

<html>
  <head>
    <title>Express app</title>
  </head>
```



```
  <body>
    <p>¡Hola Mundo!</p>
  </body>
</html>
```

6. Crear el archivo app.js con el siguiente contenido:

```
"use strict";

//imports
const express = require("express");
const http = require("http");

//app
const app = express();

//rutas
app.get("/", function(req, res) {
  res.sendFile("index.html", {root: __dirname});
});

//listen
http.createServer(app).listen(8080, function() {
  console.log("Listening...");
});
```

7. Instalar dependencias:

```
> npm install
```

8. Iniciar la aplicación Express:

```
> npm run start-dev
```

9. Abrir el navegador.
10. Ir a http://localhost:8080.

Configuración de Handlebars

Para comenzar, vamos a configurar Handlebars como motor de plantillas de la aplicación, usando como extensión de plantillas .hbs:

1. Editar el archivo package.json.
2. Añadir el paquete hbs como dependencia:

```
"dependencies": {
  "express": "*",
  "hbs": "*"
}
```

3. Guardar cambios.
4. Abrir una nueva consola.
5. Ir al directorio de la práctica.
6. Instalar la nueva dependencia:

```
> npm update
```

7. Crear el directorio de plantillas, views.
8. Editar el archivo app.js.
9. Importar Handlebars y el módulo path:

```
//imports
const express = require("express");
const http = require("http");
const path = require("path");
const hbs = require("hbs");
```

10. Configurar Handlebars:


```
//config motor de plantillas
app.set("views", path.join(__dirname, "views"));
app.set("view engine", "hbs");
app.engine("hbs", hbs.__express);
```

11. Guardar cambios.

Creación de plantillas

En este punto, ya podemos comenzar a jugar con Handlebars creando y usando plantillas:

1. Crear el archivo views/band.hbs:

```
<!doctype html>

<html>
  <head>
    <title>{{title}}</title>
  </head>

  <body>
    <p>
      Banda: {{scope.name}}<br>
      Origen:  {{scope.origin}}<br>
      Año: {{scope.year}}
    </p>
  </body>
</html>
```

   Analicemos la plantilla. Consiste en una plantilla cuyo resultado será un documento HTML. El contexto contendrá dos variables locales: title, para el título de la página; y scope, con el objeto de datos a presentar.

2. Editar el archivo app.js.

3. Añadir la constante de datos BANDS, para simular una supuesta base de datos:

```
const BANDS = {
  "vampire-weekend": {
    name: "Vampire Weekend",
    origin: "New York City, NY, US",
    year: 2006
  },
  "brmc": {
    name: "Black Rebel Motorcycle Club",
    origin: "Riverside, California, US",
    year: 1998
  }
};
```

4. Registrar la ruta /band/:name para que reproduzca la plantilla band.hbs:

```
//rutas
app.get("/", function(req, res) {
  res.sendFile("index.html", {root: __dirname});
});

app.get("/band/:name", function(req, res) {
  res.render("band", {title: "Info de banda", scope: BANDS[req.params.name]});
});
```

   Observe que cuando se solicita el procesamiento de la plantilla, no se especifica su extensión.

5. Guardar cambios.

6. Ir al navegador.

7. Solicitar http://localhost:8080/band/brmc.

8. Solicitar http://localhost:8080/band/vampire-weekend.

Definición de la plantilla de diseño

Ahora, vamos a crear la plantilla de diseño:

1. Crear la plantilla view/layout.hbs:

```
<!doctype html>

<html>
  <head>
    <title>{{title}} - Layout</title>
  </head>

  <body>
    {{{body}}}
  </body>
</html>
```

   Observe que la plantilla de diseño puede acceder al contexto de reproducción. El hueco donde se ubicará la reproducción de las plantillas se indica mediante {{{body}}}.

2. Modificar la plantilla view/band.hbs:

```
<p>
  Banda: {{scope.name}}<br>
  Origen:  {{scope.origin}}<br>
  Año: {{scope.year}}
</p>
```

3. Ir al navegador.
4. Solicitar http://localhost:8080/band/brmc.

   Comprobar que aparece Layout en el título de la página. Demostrando así que se usa la plantilla de diseño.

5. Solicitar http://localhost:8080/band/vampire-weekend.

Uso de bloques

Veamos cómo crear una plantilla que enumere el contenido de BANDS:

1. Crear la plantilla view/bands.hbs:

```
{{#each scope as |band key|}}
  {{key}}: {{band.name}} @ {{band.origin}}<br>
{{/each}}
```

   No olvidar que el primer parámetro de la cláusula as es el objeto de iteración y el segundo su clave.

2. Editar el archivo app.js.
3. Añadir la ruta /bands/:

```
//rutas
app.get("/", function(req, res) {
  res.sendFile("index.html", {root: __dirname});
});

app.get("/bands", function(req, res) {
  res.render("bands", {title: "Listado de bandas", scope: BANDS});
});

app.get("/band/:name", function(req, res) {
  res.render("band", {title: "Info de banda", scope: BANDS[req.params.name]});
});
```

4. Guardar cambios.
5. Ir al navegador.
6. Solicitar http://localhost:8080/bands.

Uso de parciales

A continuación, vamos a utilizar parciales lo que permitiría, en una aplicación más grande, reutilizar plantillas de manera muy sencilla a lo largo y ancho de la aplicación:

1. Crear el directorio views/partials.
2. Crear la plantilla views/partials/band.hbs:

```
<p>
  Banda: {{name}}<br>
  Origen:  {{origin}}<br>
  Año: {{year}}
</p>
```

3. Modificar el archivo views/bands.hbs para que use el parcial band:

```
{{#each scope as |band key|}}
  {{> band}}
{{/each}}
```

   El parcial se ejecutará bajo el contexto del elemento que la incluye.
4. Editar el archivo app.js.
5. Registrar las plantillas del directorio views/partials:

```
//parciales
hbs.registerPartials(path.join(__dirname, "views/partials/"));
```

6. Guardar cambios.
7. Ir al navegador.
8. Solicitar http://localhost:8080/bands.

Definición de funciones asistentes en línea

Y ahora, vamos a definir y usar una función asistente en línea para demostrar cómo se extiende la funcionalidad de Handlebars:

1. Modificar el archivo view/partials/band.hbs para que muestre el nombre de las bandas en mayúsculas mediante la función asistente upper:

```
<p>
  Banda: {{upper name}}<br>
  Origen:  {{origin}}<br>
  Año: {{year}}
</p>
```

2. Editar el archivo app.js.
3. Registrar la función ayudante upper:

```
//helpers
hbs.registerHelper("upper", function(text) {
  return text.toUpperCase();
});
```

4. Guardar cambios.
5. Ir al navegador.
6. Solicitar http://localhost:8080/bands.

Definición de funciones asistentes de bloque

A continuación, definamos una función asistente en bloque que muestre el nombre de una banda en mayúsculas, sólo si cumple un determinado patrón. A esta función, la llamaremos like y debe recibir como primer parámetro el valor a comparar y como segundo el patrón.

1. Modificar el archivo view/partials/band.hbs como sigue:

```
<p>
```

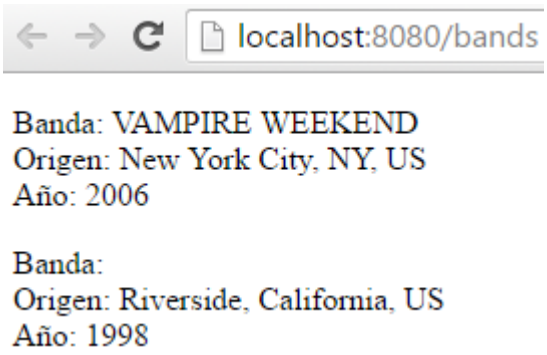


```
  Banda: {{#like name 'week'}}{{upper name}}{{/like}}<br>
  Origen:  {{origin}}<br>
  Año: {{year}}
</p>
```

2. Editar el archivo app.js.
3. Añadir la función asistente:

```
hbs.registerHelper("like", function(value, pattern, opts) {
  return (new RegExp(pattern, "i")).test(value) ? opts.fn(this) : opts.inverse(this);
});
```

4. Ir al navegador.
5. Solicitar http://localhost:8080/bands:



Observe que no aparece el nombre de la segunda banda. Normal. La función asistente sólo procesa su cuerpo si el nombre cumple la expresión regular. En otro caso, no. De ahí que sólo se muestre el nombre de la primera.

6. Modificar el parcial views/partials/band.hbs:

```
<p>
  Banda: {{#like name 'week'}}{{upper name}}{{else}}{{name}}{{/like}}<br>
  Origen:  {{origin}}<br>
  Año: {{year}}
</p>
```

Observe la aparición de la cláusula {{else}}. Contiene lo que debe devolver opts.inverse(this). Si no se indica, no devolverá nada. Pero si se indica, devolverá su cuerpo asociado.

7. Ir al navegador.
8. Solicitar http://localhost:8080/bands: