Universidad Politécnica Salesiana

Ingeniería en Ciencias de la Computación

WEB SERVER

Informe 03

Autor:
Ricardo Romo

June 24, 2020

0.1 Requisitos

Para crear un web server lo primero que necesitaremos son los módulos express, para arrancar el servidor, **handlebars** para los parciales y helpers

1. **Peraciales**: Templates o plantillas para los encabezados de las paginas html, a demás son el medio por las que se envían los datos
2. **Helper**: Son funciones o ejecuciones del programa que se necesita corran en ciertas áreas cuando son llamadas

Instalacion Librerias:

1. Iniciar npm con el comando **npm**
2. Instalar la librería **express** con **npm install express –save**
3. Instalar el módulo **handlebars** con **npm install hbs –save**
4. Crear una carpeta llamada **hbs** y dentro de esta un documento llamado **helpers.js** los cuales almacenaran nuestros helpers para realizar funciones.
5. Crear una carpeta **public** que es donde se almacenan a nuestros archivos estáticos que necesitara express
6. Crear una carpeta llamada **views** y dentro de esta una carpeta llamada **parciales** y un archivo llamado **about.hbs** y uno llamado **home.hbs** que son nuestras páginas home y about con formato html
7. Dentro de la carpeta parciales crear los archivos **footer.hbs** y **header.hbs** que son templates de las cabecera y pie de página de nuestras páginas html.

Express

Handlebars

0.2 Server.js

1. Para iniciar primero se instancian los módulos de **express** y hbs además se guarda el módulo de express dentro de una variable llamada **app**

```
1  const express = require('express')
2  const app = express()
```

2. Se inicia la librería de hbs con require

3. A demás de eso llamamos al puerto disponible que se nos asigna para poder levantar eln servidor

4. Llamamos a nuestro archivo donde se almacenaran los helpers

```
1  const hbs = require('hbs');
2  const puerto = process.env.PORT || 3000
3  require('./hbs/helpers');
```

5. Utilizamos el comando **use** para establecer donde esta nuestra carpeta con nuestros archivos estáticos, además con el comando **__dirname** establecemos la ruta de ejecución del servidor.

6. Utilizamos el comando **set** para establecer el manejo de los archivos html y especificamos que es con **hbs**

7. el comando **hbs.registerPartials** nos ayuda a establecer donde almacenaremos nuestros templates.

```
1  app.use(express.static(__dirname + '/public'))
2  app.set('view engine', 'hbs');
3  hbs.registerPartials(__dirname + "/views/parciales");
```

8. Con el comando **app.get** establecemos un pedido get a nuestro servidor, además con se debe de especificar la dirección a donde hacemos la petición, en este caso especifica la dirección raíz,

9. con **res.render** establecemos los datos que deseamos enviar en este caso se enviara a nuestro archivo **home.hbs** los datos del nombre del usuario, la página y activación(pestañas en el navbar activables)

```
1  app.get('/', (req, res) => {
2      res.render('home', {
3          nombre: 'rICArDo rOmO',
4          pagina: "home",
5          activacion: { h: "active", a: "" }
6      });
7  })
```

10. Hacemos otra petición get pero con la dirección **about**

11. Esta llamara a nuestro **about.hbs** y enviara los datos de página y que pestañas del navbar serán activables

```
app.get('/about', (req, res) => {
    res.render('about', {
        pagina: "about",
        activacion: { h: "", a: "active" }
    })
});
```

12. **app.listen()** levanta el puerto donde se dará el servicio.

```
app.listen(puerto, () => {
    console.log(`Escuchando en el puerto , http://
        localhost:${puerto}`);
})
```

0.3 helpers.js

Ingresamos dentro del archivo **helpers.js** dentro de nuestra carpeta **hbs**

1. se llama al módulo **hbs** antes ya instalado

```
const hbs = require('hbs')
```

2. Con el comando **hbs.registerHelper()** se registra una función que se ejecutara en el html ('frontend') el cual recibe los parámetros del nombre de este helper y retorna un resultado

3. En este caso nuestro herlper se llama **getAnio** y retorna el parámetro del año actual que uno se encuentra.

```
hbs.registerHelper('getAnio', () => {
    return new Date().getFullYear()
```

4. Ahora se registraremos un nuevo template llamado **capitalizar** el cual recibirá un **string** y se encargara de dar un formato a cada palabra de esta, devolviendo el primer carácter de la palabra en mayúscula y los demás en minúscula

```
hbs.registerHelper('capitalizar', (texto) => {
    let palabras = texto.split(' ')
    palabras.forEach((palabra, idx) => {
        palabras[idx] = palabra.charAt(0).toUpperCase() +
            palabra.slice(1).toLowerCase()
    });
    return palabras.join(' ');
})
```

0.4 Parciales

Debido a q muchas páginas web utilizan los mismos encabezados, los parciales nos ayudan a tener cabeceras y pies de páginas ya estáticos y fáciles de llamar

0.4.1 header.hbs

Dentro del header tendremos nuestra cabecera la cual será el inicio del documento

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
    <title>{{pagina}}|Demo</title>
    <link rel="stylesheet" href="https://bootswatch.com/4/lux/
        bootstrap.css">
    <!-- <link rel="stylesheet" href="https://bootswatch.com/
        _assets/css/custom.min.css"> -->
</head>

<body>
```

en nuestra cabecera podemos observar que para él la etiqueta del título llamamos al atributo **pagina** el cual se envía a través del server en la **petición get a nuestra dirección raíz**
Nota: Además es importante recalcar que no estamos usando **bootstap** al contrario de esto estamos usando un derivado de este que **bootswatch** diseño **lux** el cual es amigable al llamar a través de un solo link y sin necesidad de crear carpetas secundarias con estilos https://bootswatch.com/lux/
Url guía componentes: https://bootswatch.com/lux/
url para estilos css: https://bootswatch.com/4/lux/bootstrap.css

0.4.2 footer.hbs

Dentro de **footer.hbs** tendremos lo que vendría siendo nuestro pie de pagina

```
    <footer class="text-center">
        Universidad Politecnica  <br/>
        Ricardo Romo-{{getAnio}}
    </footer>
    <style type="text/css">
    footer {
  background-color: black;
  position: absolute;
  bottom: 0;
  width: 100%;
  height: 40px;
  color: white;
}
```

```
    </style>
</body>

</html>
```

Nota: Se puede observar cómo se llamar al helper **getAnio** el cual nos regresa nuestro año actual

0.4.3 navbar.hbs

Aquí instanciamos lo que vendría siendo la barra de opciones de nuestra página, este nav bar fue copiado de bootswatch el cual ya nos da con su diseño exclusivo. Además, podemos notar que enviamos las variables de activación para saber cuál pestaña u opción deberá prenderse a al estar en cierta página.

```
<div>
        <nav class="navbar navbar-expand-lg navbar-dark
            bg-primary">
            <a class="navbar-brand" href="#">Navbar</a>
            <button class="navbar-toggler" type="button"
                data-toggle="collapse" data-target="#
                navbarColor01" aria-controls="navbarColor01"
                aria-expanded="false" aria-label="Toggle
                navigation">
              <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="
                navbarColor01">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item {{activacion.h}}">
                        <a class="nav-link" href="/">Home <span
                             class="sr-only">(current)</span></
                            a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Features</
                            a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Pricing</a
                            >
                    </li>
                    <li class="nav-item {{activacion.a}}">
                        <a class="nav-link " href="about">
                            About</a>
                    </li>
                </ul>
                <form class="form-inline my-2 my-lg-0">
                    <input class="form-control mr-sm-2" type="
                        text" placeholder="Search">
                    <button class="btn btn-secondary my-2
                        my-sm-0" type="submit">Search</button>
                </form>
            </div>
        </nav>
```

0.4.4 home.hbs

Esta página vendría siendo nuestra página principal al arrancar nuestro servidor y mostrarse en el navegador.

```
<div>
        <nav class="navbar navbar-expand-lg navbar-dark
            bg-primary">
            <a class="navbar-brand" href="#">Navbar</a>
            <button class="navbar-toggler" type="button"
                data-toggle="collapse" data-target="#
                navbarColor01" aria-controls="navbarColor01"
                aria-expanded="false" aria-label="Toggle
                navigation">
              <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="
                navbarColor01">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item {{activacion.h}}">
                        <a class="nav-link" href="/">Home <span
                             class="sr-only">(current)</span></
                            a>
                    </li>
                    <li class="nav-item">
```

Nota: Podemos apreciar como llamamos a nuestros templetes o parciales como nuestro **header,navbar y footer**, además podemos observar cómo llamamos a nuestro helper **capitalizar** y le enviamos el parámetro de **nombre** antes ya enviado por nuestro **server.js** a través de una petición get.

0.4.5 about.js

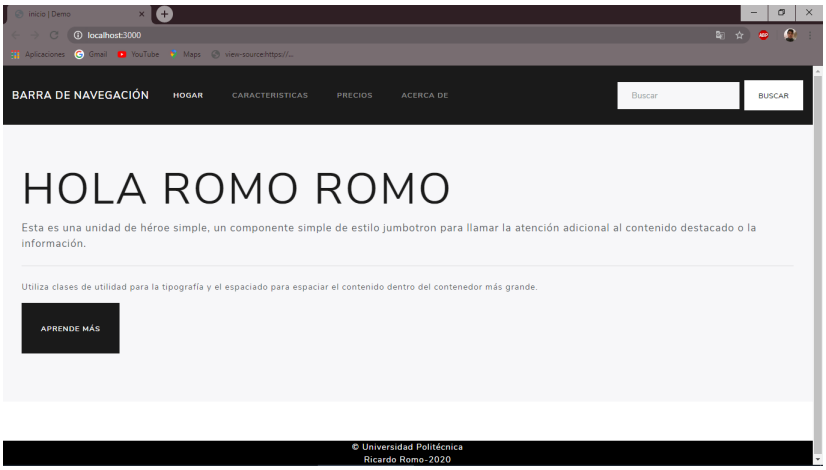
Así mismo podeos observar nuestro **about** el cual así mismo llamaremos a nuestros parciales o templetes para optimizar líneas de código html.

```
{{> header}}
{{> navbar}}
<div>
    <h1>
        MI PAGINA CON WEB SERVICE
    </h1>
</div>
{{> footer}}
```

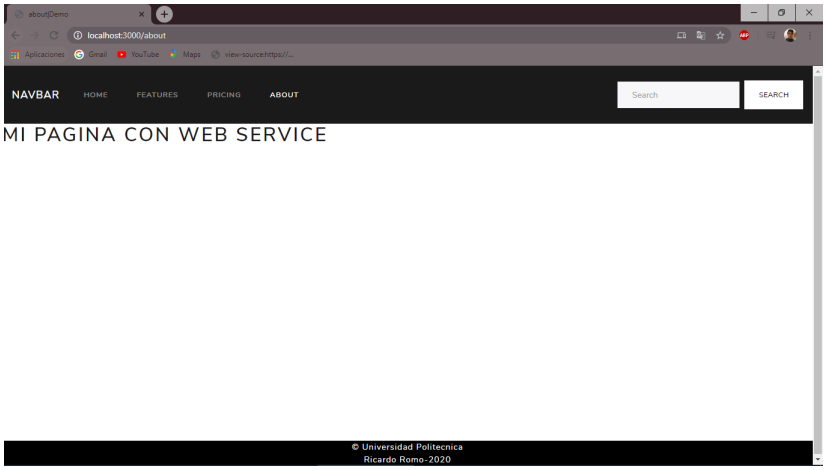
0.5 Ejecucion

1. Primero ejecutamos el comando **node server.js**

2. Luego nos dirijimos dentro del navegador a la direccion que nos muestra como resultado



3. Al dar click en abaout podemos redirigirnos a la ventana about.hbs



4. Y lo mismo al retornar al home

0.6 Repositorio

github: https://github.com/rromom/clases-web/tree/master/codigos/web_server

heroku: https://romo-web-service.herokuapp.com/