UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

NodeJS Nivel Intermedio

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 1

Unidad 3: View Engine

Presentación:

Actualmente NodeJS es una de las principales tecnologías de desarrollo del lado servidor. Es utilizado por empresas como Netflix, PayPal, LinkedIn, Uber, Walmart, EBay entre otras. Una de las características más atractivas de este runtime es que se basa en JavaScript por lo que los desarrolladores front-end pueden desarrollar back-end sin necesidad de aprender un nuevo lenguaje de programación.

En esta unidad introduciremos el concepto de plantillas y motores de plantillas. Luego, abordaremos la instalación del motor de plantillas Handlebars, uno de los más usados del mercado, y su integración con Express. Para finalizar, revisaremos la estructura básica de Handlebars una vez integrado con Express, lo que nos permitirá comenzar a trabajar con esta valiosa herramienta para la realización de aplicaciones NodeJs.

Objetivos:

Que los participantes*:

- Conozcan cómo utilizar motores de templates en Express
- Sepan cómo trabaja el motor de templates Handlebars

Bloques temáticos*:

1. Server Side Rendering
    - ¿Qué es una plantilla HTML?
    - Administración de plantillas
2. Handlebars
    - ¿Qué es Handlebars?
    - Instalación e integración con Express
3. Ruteo
    - Cómo realizar el ruteo
4. Modularización
    - Cómo modularizar Handlebars

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

** El MEC es el modelo de E-learning colaborativo de nuestro Centro.*

Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

**** Está página queda como está. El contenidista no le quita ni le agrega nada.***

1. Server Side Rendering

¿Qué es una plantilla HTML?

Una plantilla es un modelo escrito en código HTML (el lenguaje que interpretan los navegadores web) que se utiliza como base para crear nuevas páginas web. Una maqueta, un "molde" que permite realizar páginas web de forma sencilla, homogénea y rápida. Cuando se genera una **página web basada en una plantilla**, la página web **tendrá similar diseño pero distinto contenido**.

Cada persona puede generar sus propias plantillas o bien utilizar algunas ya existentes. De hecho, existen sitios completos destinados a ofrecer plantillas HTML!

Los sistemas web también se benefician con el uso de plantillas y la forma de utilización es similar a la que se implementa en los sitios web, es decir, se utiliza la plantilla para generar otras páginas con similar diseño y contenido distinto.

Durante el desarrollo de un sistema web, se utilizará una o varias plantillas que se combinarán con los datos entregados por la lógica de programación para así enviar al navegador, una página web con diseño similar (lo que otorga homogeneidad al usuario, brindándole una mejor experiencia), y la información solicitada por dicho usuario.

Ejemplo de porción de código de una posible plantilla HTML

```
<div class="estructura">
        <h1>{{título}}</h1>
        <div class="cuerpo">
                {{cuerpo}}
        </div>
</div>
```

Como vemos en este ejemplo, *{{título}}* es el "lugar" donde iría el título de esta página, similar para *{{cuerpo}}*.

Al utilizar esta plantilla, deberíamos copiar este código y reescribir los lugares indicados con la información particular del caso.

En el mundo de la programación, los “lugares” como {{título}} y {{cuerpo}} se denominan **marcadores**. Mantengamos esto en mente para futuras menciones.

Administración de plantillas

Como se mencionó líneas arriba, el uso de plantillas se ha generalizado en el mundo de las aplicaciones web. Ellas colaboran con la separación entre la lógica de programación y la presentación de la información.

Pero claro, en el caso de un sistema web, se requiere automatización en el proceso de tomar la plantilla, tomar los datos y generar las nuevas páginas web que luego serán entregadas al sistema que le solicitó el trabajo. Esta automatización se realiza mediante *motores de plantillas*.

Los *motores de plantillas* son justamente eso: programas que conocen un tipo de plantillas, escritas en un lenguaje en particular y que pueden mediante petición, generar páginas web con similar diseño al de la plantilla e información diferente (la enviada por el sistema) para ser entregadas al sistema que las solicitó.

Existen muchos motores de plantillas, cada desarrollador tiene la libertad de trabajar con el que desea. Lo cierto es que cuando un desarrollador se acostumbra a trabajar con un motor, intentará utilizarlo siempre, ya que de esta forma ahorrará mucho tiempo de aprendizaje. Siendo así, podemos tener una idea de la importancia de elegir para aprender y trabajar un motor popular.

Express trae incorporado para la administración de plantillas el motor Jade pero por supuesto, permite que el desarrollador elija otro motor para trabajar. En este curso, nosotros elegiremos trabajar con Handlebars, que es un motor de plantillas basado en JavaScript, útil tanto para el lado cliente como para el lado servidor y que además es muy popular entre los desarrolladores.

Esto último no es un dato menor! Cuánto más popularidad y adeptos gane un motor, más soporte tendrá en los foros de discusión on-line así que cuando surja algún error o duda sobre su funcionamiento se tendrá dónde consultar y además habrá más oportunidades laborales para aquellos que lo conozcan ya que más empresas lo implementarán.

2. Handlebars

¿Que es Handlebars?

Handlebar es un motor de plantillas. Es una extensión de Mustache.js (otro motor de plantillas) por lo que la sintaxis de Handlebar es compatible con la de Mustache.js.

Handlebar está basado en JavaScript y se puede utilizar tanto del lado cliente como del lado servidor. En este curso lo trabajaremos del lado servidor, ya que enviaremos como respuesta, la totalidad de la página web que el usuario verá.

Recordemos el ejemplo anterior….

Nombre de la plantilla: ejemploPlantilla.hbs

```
<div class="estructura">
<h1>{{título}}</h1>
        <div class="cuerpo">
                {{cuerpo}}
        </div>
</div>
```

El uso de un motor de plantillas, agilizaría el proceso de creación de nuevas páginas en base a esta plantilla. Reemplazaría automáticamente los campos {{título}} y {{cuerpo}} por lo que arroje la lógica de programación.

Un circuito posible sería:

1. El sistema solicita una página al motor de plantilla enviándole la siguiente información:

> Nombre de la plantilla a utilizar: ejemploPlantilla.hbs
> título:"Mi primer plantilla"
> cuerpo:"Hola, estoy usando por primera vez una plantilla!"

2. El motor busca ejemploPlantilla.hbs y genera un archivo HTML "ejemploPlantilla.html" que contiene el mismo código de la plantilla pero sustituyendo los campos título y cuerpo por lo recibido desde el sistema

3. Devuelve al sistema, la página creada

Aquí queda bien en claro el beneficio mayor de la utilización de un motor de plantillas, ya que al automatizar la creación de las páginas no se corre el riesgo de corromper el código HTML al editar los campos, lo que sí puede suceder al hacerlo manualmente.

Instalación

Para realizar la instalación seguir los siguientes pasos:

1. En la consola

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

2. En el archivo de configuración de Express agregar lo siguiente:

```
var handlebars = require('express-handlebars') . create({defaultLayout: 'main' });
app.engine('handlebars', handlebars.engine);app.set('view engine', 'handlebars');
```

Listo! La próxima vez que iniciemos Express ya podremos trabajar con Handlebars.

Integración

Una vez instalado Handlebars también se ha realizado la integración con Express de forma automática.

Dicha integración entre otras cosas muy importantes por cierto pero que escapan a este curso, creará una estructura de carpetas y algunos archivos base para que el desarrollador pueda tomar de ejemplo.

La estructura básica de carpetas de handlebars será más o menos así:

```
.
├── app.js
└── views
    ├── home.handlebars
    └── layouts
        └── main.handlebars
```

Como se ve en esta imágen, algunos archivos y carpetas permitirán trabajar cómodamente con handlebars.

En lo posible es conveniente mantener esta estructura, al menos la iniciar el recorrido de aprendizaje ya que aún no se cuenta con los conocimientos necesarios para poder realizar todas las modificaciones de configuración requeridas para que otra estructura resulte compatible.

3. Ruteo

Cómo realizar el ruteo

Una ruta es un camino, una vía por la cual llegar a cierto lugar. Esto también en cierto para nosotros los desarrolladores!

Por ahora no conocemos algunos conceptos necesarios para entender en profundidad la dimensión de una ruta y el enrutamiento pero podemos desde lo pragmático, aprender cómo opera Express con Handlebars para poder entregar una página en particular que haya sido solicitada.

Para arrancar este recorrido, necesitamos conocer un poco más acerca de los archivos...

app.js

Esta aplicación en Express muy simple que se crea, contiene un modo básico de registrar un motor de Handlebars.

```
var express = require('express');
var exphbs  = require('express-handlebars');

var app = express();

app.engine('handlebars', exphbs({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

app.get('/', function (req, res) {
    res.render('home');
});

app.listen(3000);
```

views/layouts/main.handlebars

Este archivo contiene un diseño básico en HTML que se puede utilizar de base para crear las diferentes vistas de la app.

El marcador {{{body}}} indica el lugar donde se reproducirá el contenido principal.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example App</title>
</head>
<body>

    {{{body}}}

</body>
</html>
```

views/home.handlebars

En este archivo con extensión “handlebars” se encuentra un modelo de contenido de vista que se reproducirá en lugar del marcador {{{body}}} del archivo views/layouts/main.handlebars.

Como se aprecia, el archivo main.handlebars trabaja en conjunto con el archivo home.handlebars para generar la página HTML final que como se vio en secciones anteriores, contendrá similar diseño que main.handlebars pero distinto contenido (el que tomará de home.handlebars).

```
<h1>Example App: Home</h1>
```

Vamos a retomar el archivo app.js porque allí se encuentra una parte importante del concepto de enrutamiento de Handlebars integrado con Express.

```
var express = require('express');
var exphbs  = require('express-handlebars');

var app = express();

app.engine('handlebars', exphbs({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

app.get('/', function (req, res) {
    res.render('home');
});

app.listen(3000);
```

Lo analizaremos por partes:

```
app.get('/', function (req, res) {
    res.render('home');
});
```

En un lenguaje coloquial, estas líneas podrían traducirse como:

*Si "alguien" pregunta por **'/'**, entonces responderás enviándole lo que encuentres en la carpeta 'views' con nombre **'home'**.*

Seguramente el lector se pregunte por qué irá a buscar 'home' dentro de la carpeta 'views' aunque no esté indicado en el código explícitamente? Bien, al momento de la integración entre Handlebars y Express, archivos de configuración han sido alterados para setear estas y otras cuestiones.

Estas líneas que analizamos forman parte del proceso de enrutamiento, ya que se indica un camino (en este caso **'/'**) y un destino (**'home'**).

Del mismo modo, será posible indicar otros caminos (rutas) y destinos. Esto se completa con otras partes importantes del código de app.js que veremos enseguida.

Continuemos:

```
app.engine('handlebars', exphbs({defaultLayout: 'main'}));
```

El archivo main posee el diseño y el archivo home el contenido a fusionar con el diseño. En esta línea se indica explícitamente que el diseño a utilizar será el que lleve por nombre “main”.

Siendo así el proceso completo y conforme el código de app.js cuando “alguien” pregunte por ‘/’ la aplicación tomará el contenido del archivo ‘home’ y lo incrustará en el diseño del archivo ‘main’, justo donde encuentre el marcador {{{body}}} (otra de las cosas que se encuentran indicadas en la configuración general). Con todo lo anterior, creará un archivo HTML que pondrá a disposición de la aplicación.

4. Modularización

Cómo modularizar Handlebars

Aquí nos referimos a modularizar como ordenar los distintos archivos que son necesarios para que Handlebars funcione con Express.

Las carpetas que se crean dividen los diferentes componentes permitiendo ordenar y facilitar el uso de los mismos.

Retomemos la estructura de carpetas:

```
.
├── app.js
└── views
    ├── home.handlebars
    └── layouts
        └── main.handlebars
```

Así el archivo app.js se encuentra en la carpeta raíz, luego el archivo home.handlebars dentro de la carpeta views (vistas en inglés) lo que nos brinda una forma intuitiva de saber

qué tipo de contenido debemos guardar aquí. Del mismo modo, la carpeta layouts dentro de la carpeta views, que contiene los modelos HTML como main.handlebars.

A modo de conclusión se observa que al integrar Handlebars con Express, el primero adiciona un conjunto de archivos base y algunos directorios que facilitarán el orden en el trabajo con esta tecnología.

En cuanto a los archivos, el app.js muestra un ejemplo del código básico que debe ser integrado a cualquier aplicación en Express que quiera trabajar con handlebars.

El archivo main.handlebars dentro de la carpeta views/layouts/ contiene un ejemplo de diseño de plantilla. Toda plantilla que se cree para el sistema a desarrollar deberá ir en esta carpeta y seguir la estructura propuesta por el archivo main.handlebars.

Finalmente el archivo home.handlebars muestra muy sintéticamente la forma en que se deben crear los archivos de vistas que se integrarán a las plantillas. Toda vista, deberá ser creada en este directorio.

Bibliografía utilizada y sugerida

Node Package Manager Recuperado de https://www.npmjs.com/package/express-handlebars

Sitio oficial de Handlebars Recuperado de http://handlebarsjs.com/

Lo que vimos:

Server Side Rendering

Handlebars

Ruteo

Modularización

Lo que viene:

API. Conceptos básicos

Respuesta JSON

Ruteo