<codoacodo/>

Agencia de Aprendizaje a lo largo de la vida

Desarrollo Fullstack

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

<codoacodo/>

Clase 33

Workshop

- Integración de las capas MVC vistas hasta el momento.

Clase 34

Node JS

- Capa de Vistas
- Motores de Plantillas
- Sintaxis EJS

Clase 35

Node JS

- Multer
- Express Validator

NODE JS

MVC + Template Engines

Agencia de Aprendizaje a lo largo de la vida

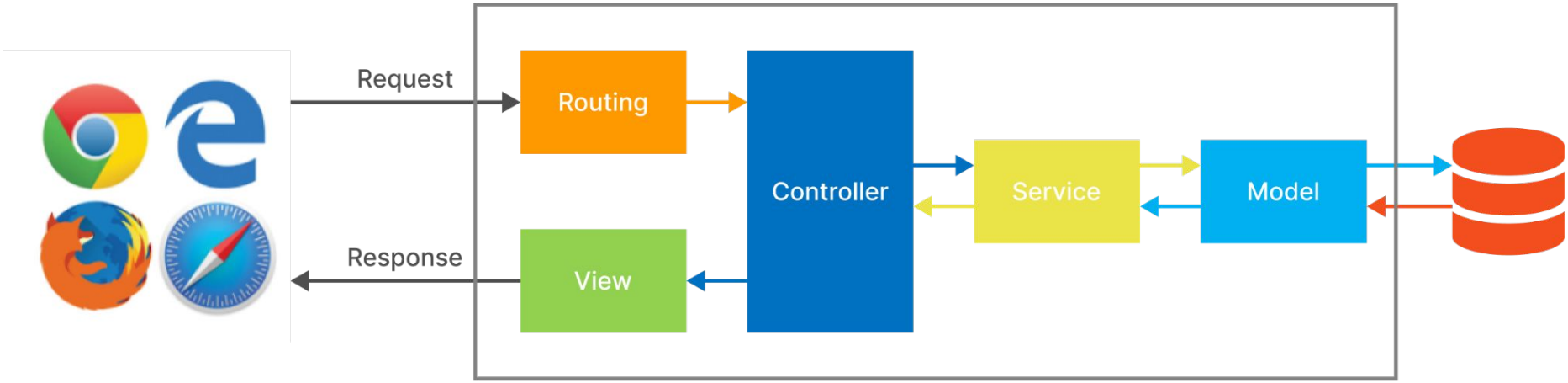
Buenos Aires Ciudad Educación

Hagamos repaso...

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

MVC + Services



Vista

Es la capa visible de nuestra aplicación.

Es el código ***HTML***, ***CSS***, ***Javascript*** necesario para renderizar y mostrar los datos e información a nuestros usuarios.

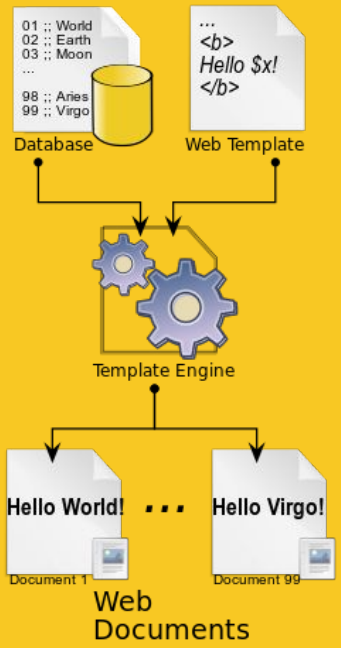
En ocasiones se suelen utilizar ***Template Engines*** o diversos frameworks como ***React*** para adoptar flexibilidad en el desarrollo.

Normalmente es la capa Front End de los proyectos y su comunicación hacia las fuentes de datos se realiza a través de los ***controladores***.

Template Engines

Los motores de plantillas nos permiten trabajar nuestras vistas mediante contenido dinámico.

Primero marcamos la estructura con HTML y luego enviamos datos que serán manejados con javascript de forma variable.



Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

Template Engines

Existen una gran cantidad y para **diversos lenguajes**, **Express** detalla cuales soporta de forma nativa.

These template engines work "out-of-the-box" with Express:

- **Pug**: Haml-inspired template engine (formerly Jade).
- **Haml.js**: Haml implementation.
- **EJS**: Embedded JavaScript template engine.
- **hbs**: Adapter for Handlebars.js, an extension of Mustache.js template engine.
- **Squirrelly**: Blazing-fast template engine that supports partials, helpers, custom tags, filters, and caching. Not white-space sensitive, works with any language.
- **Eta**: Super-fast lightweight embedded JS template engine. Supports custom delimiters, async, whitespace control, partials, caching, plugins.
- **combyne.js**: A template engine that hopefully works the way you'd expect.
- **Nunjucks**: Inspired by jinja/twig.
- **marko**: A fast and lightweight HTML-based templating engine that compiles templates to CommonJS modules and supports streaming, async rendering and custom tags. (Renders directly to the HTTP response stream).
- **whiskers**: Small, fast, mustachioed.
- **Blade**: HTML Template Compiler, inspired by Jade & Haml.
- **Haml-Coffee**: Haml templates where you can write inline CoffeeScript.
- **express-hbs**: Handlebars with layouts, partials and blocks for express 3 from Barc.
- **express-handlebars**: A Handlebars view engine for Express which doesn't suck.
- **express-views-dom**: A DOM view engine for Express.
- **rivets-server**: Render Rivets.js templates on the server.
- **LiquidJS**: A simple, expressive and safe template engine.
- **express-tl**: A template-literal engine implementation for Express.
- **Twing**: First-class Twig engine for Node.js.
- **Sprightly**: A very light-weight JS template engine (45 lines of code), that consists of all the bare-bones features that you want to see in a template engine.
- **html-express-js**: A small template engine for those that want to just serve static or dynamic HTML pages using native JavaScript.

Si bien hay bastantes, los más conocidos son:

<%= EJS %>

JSX

pug

handlebars

NUNJUCKS

Ventajas

Ayuda a la organización del proyecto.

Permite el uso de Javascript dentro del mismo código HTML a través de la sintaxis del template engine elegido.

No tenemos que tener archivos estáticos para cada tipo de respuesta, las vistas son dinámicas.

Crear templates para no repetir bloques de vistas es muy sencillo.

Instalación de EJS

Corremos el comando: `npm install ejs`

Luego indicamos en nuestro entry point que usaremos un motor de plantillas y donde alojaremos los templates:

```
/* Motor de Plantillas EJS */
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname,
'./src/views'));
```

Ahora ya podemos crear nuestras primeras vistas.

Devolviendo una vista en EJS

Para retornar un archivo EJS en formato **HTML**, debemos utilizar el método ***render*** de **res**.

```
app.get("/", (req, res) => {
  res.render(path.resolve(__dirname, '../views/home')
});
```

*Gracias a este método, nuestro programa tomará el archivo con extensión .ejs y lo **enviará al cliente** como un archivo .html tradicional.*

Sintaxis

EJS posee su propia sintaxis.

En primer lugar nuestros archivos **.html**, tendrán la extensión **.ejs**

Si bien el código **HTML** que escribamos será 100% válido, ahora podemos agregarle algunos **super poderes**.

```
<%- include('partials/header'); -%>
<main>
    <section class="hero">
        <article class="hero__info container">
            <h3 class="hero__info--title">Nuevos Ingresos</h3>
            <p class="hero__info--desc">Descubrí el próximo Funko Pop de tu colección</p>
            <a class="hero__info--link" href="/shop?sort=last_arrived">SHOP</a>
        </article>
    </section>
    <% collections.forEach((collection, i) => { %>
        <section class="collection container">
            <article class="collection__info">
                <h3 class="collection__info--title"><%= collection.title.toUpperCase(); %
                <p class="collection__info--desc"><%= collection.description %></p>
                <a class="collection__info--link" href="/shop?filter=<%= collection.colle
            </article>
```

TAGS EJS

```
Utilizado para el control de flujo sin salida
<% collections.forEach((collection, i) => { %>
     // Código HTML
<% } %>

Devuelve un valor único como una variable
<%= collection.collection %>

Importa partes de otros HTML
<%- include('partials/header'); -%>

Comentario, no se imprime
<%# Está línea está comentada #%>
```

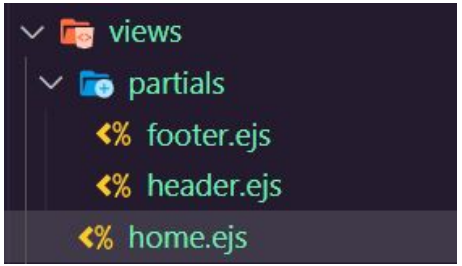
- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%_` 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<%#` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` 'Whitespace Slurping' ending tag, removes all whitespace after it

Partials

Son piezas de código HTML que usualmente se repiten en todas las vistas.

Es común crear un archivo para el header, el footer o incluso el head de nuestro sitio.

Al tener el código solo una vez, cualquier cambio realizado en estos archivos impacta en aquellos lugares donde sea requerido.




```
<%- include('partials/header') %>
<main>
    <section>
        <h2>HOME del sitio</h2>
        <p>Lorem ipsum dolor.</p>
    </section>
</main>
<%- include('partials/footer') %>
```

Con los tags `<%- %>` y la función `include()` importamos los archivos parciales que necesitemos.

Variables

También es posible enviar información a nuestro HTML

```
home: (req, res) => {
        return res.render('home', {
            message: 'Este texto es dinámico'
        });
    }
```

Para usarla luego desde nuestro archivo EJS.

```
<main>
    <section>
        <h2><%= message %></h2>
        <p>Lorem ipsum dolor.</p>
    </section>
</main>
```

Los tags `<%= %>` permiten **representar una variable** para ese segmento de código.

Bloques de código

Además de mensajes podemos enviar objetos con mucha más información, por ejemplo un array de personajes.

```
<section class="characters">
    <% characters.forEach(character => { %>
        <article class="character" >
            <div class="character">
                <a href="/character/<%= character.id %> ">
                    <h3 class="name"><%= character.name  %></h3>
                </a>
                <p class="specie"><%= character.species  %></p>
            </div>
        </article>
    <% }) %>
</section>
```

Los tags `<% %>` se utilizan para escribir bloques libres de código javascript, **debemos utilizarlos al inicio** y **al final** de nuestro código y dentro todo el HTML que necesitemos.

Si bien estos son los tags más comunes de EJS existen muchos otros, solo queda revisar la documentación.

https://ejs.co/#docs

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.

Gracias