



Clase 9. Programación Backend

Motores de Plantillas



OBJETIVOS DE LA CLASE

- Entender qué es un motor de plantillas y su implementación en el backend.
- Conocer el motor de plantillas Handlebars: sintaxis y uso, e integrarlo a Express

CRONOGRAMA DEL CURSO

Clase 8



Router & Multer

Clase 9



Motores de Plantillas

Clase 10



Pug & Ejs

Repasando... 

```

1 const { Router } = require('express')
2 const UsuariosController = require('../controllers/usuarios')
3 const validatorCrearUsuarioMiddleware = require('../middlewares/validador-crear-usuarios')
4
5 const router = Router()
6
7 router.post('/', validatorCrearUsuarioMiddleware, async (req, res, next) => {
8   try {
9     const usuario = await UsuariosController.crear(req.body)
10    res.status(201).json(usuario)
11  } catch (error) {
12    next(error)
13  }
14 })
15
16 module.exports = router

```

Router & Middlerwares

```

1 module.exports = async function validatorCrearUsuarioMiddleware(req, res, next) {
2   const logPrefix = '[validadorCrearUsuarioMiddleware]'
3   try {
4     console.log(`${logPrefix} intentando validar body usuario...`)
5     req.body = await usuarioSchema.validateAsync(req.body)
6     console.log(`${logPrefix} validación body usuario exitosa.`)
7     next()
8   } catch (error) {
9     console.error(`${logPrefix} validación fallida de body usuario: ${error.message}`)
10    next(new BadRequestError('Ocurrio un error validando', error))
11  }
12 }

```

```

1  const { Router } = require('express')
2  const multer = require('multer')
3  const {actualizarAvatarPorId} = require('../controllers/usuarios')
4  const {BadRequestError} = require('../utils/errores')
5  const validadorUsuarioExisteMiddleware = require('../middlewares/validador-usuario-existe')
6
7  const router = Router()
8
9  const storage = multer.diskStorage({
10     destination: function (req, file, cb) {
11         cb(null, 'pictures/')
12     },
13     filename: function (req, file, cb) {
14         const usuarioId = req.usuario._id
15         const extArray = file.mimetype.split('/')
16         const extension = extArray[extArray.length - 1]
17         cb(null, `${usuarioId}.${extension}`)
18     }
19 })
20
21 const upload = multer({ storage })
22
23 router.put('/:id/avatar',
24     validadorUsuarioExisteMiddleware,
25     upload.single('avatar'),
26     async (req, res, next) => {
27         try {
28             if (!req.file) {
29                 throw new BadRequestError('Debe subir una archivo valido para esta acción.')
30             }
31             const result = await actualizarAvatarPorId(req.usuario._id, req.file)
32             res.json(result)
33         } catch (error) {
34             next(error)
35         }
36     })
37
38 module.exports = router

```

Multer

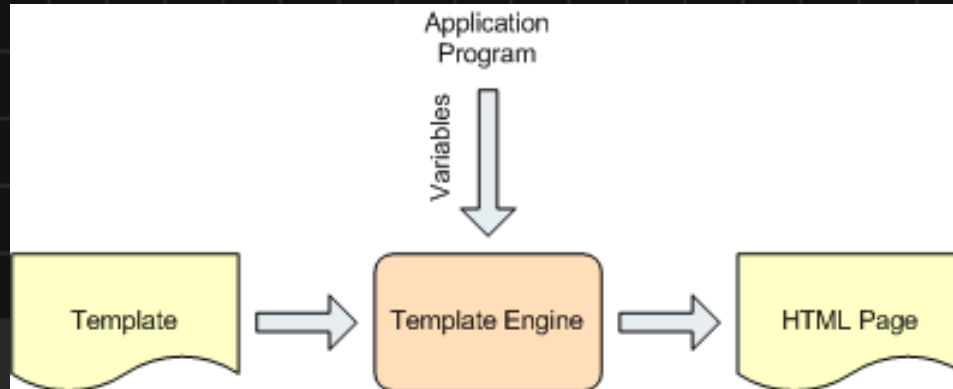
CODER HOUSE

Vamos al IDE... 



***¿Alguna pregunta hasta
ahora?***

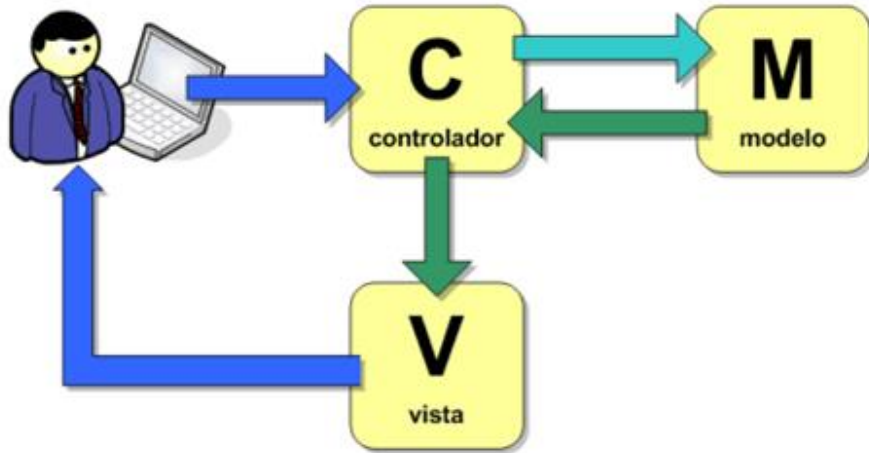
Motores de plantillas (Template engines)



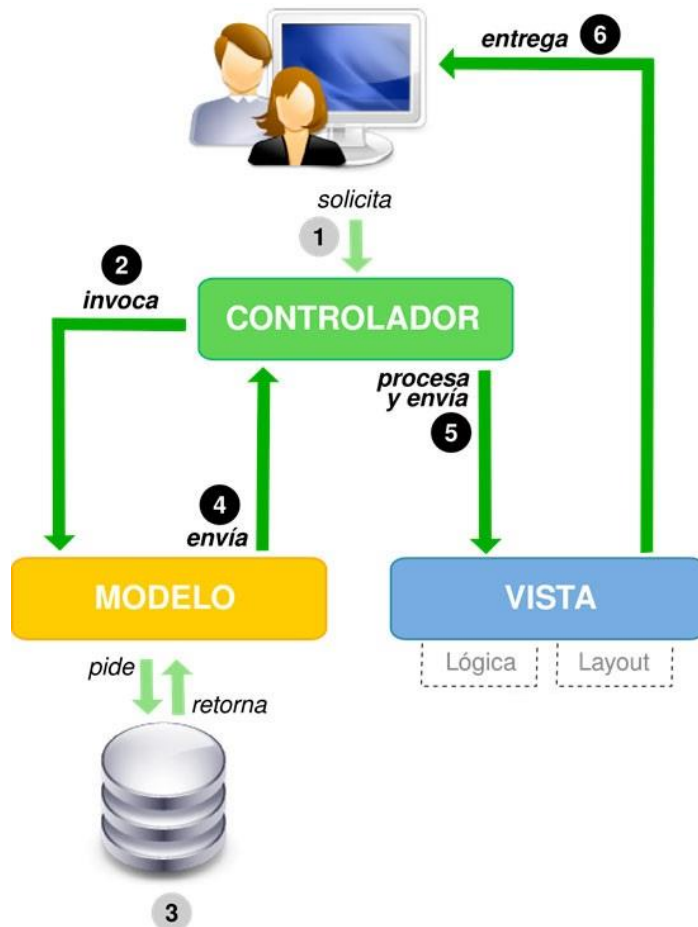
- En la programación en general y en la programación web en particular existe el denominado patrón **MVC** (*Modelo Vista Controlador*)
- Este patrón trata de **separar los datos de su presentación**. Por decirlo en términos web, separar el código del programador del código del diseñador web.
- Las plantillas (templates) son una aproximación más para resolver este problema.

Modelo Vista Controlador





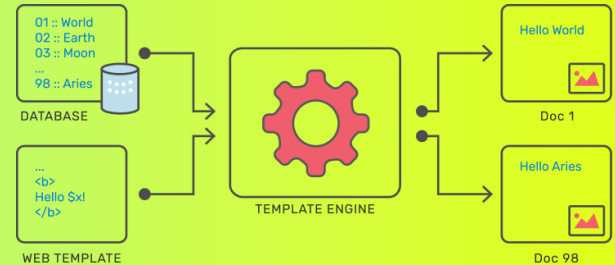
***Modelo Vista
Controlador***

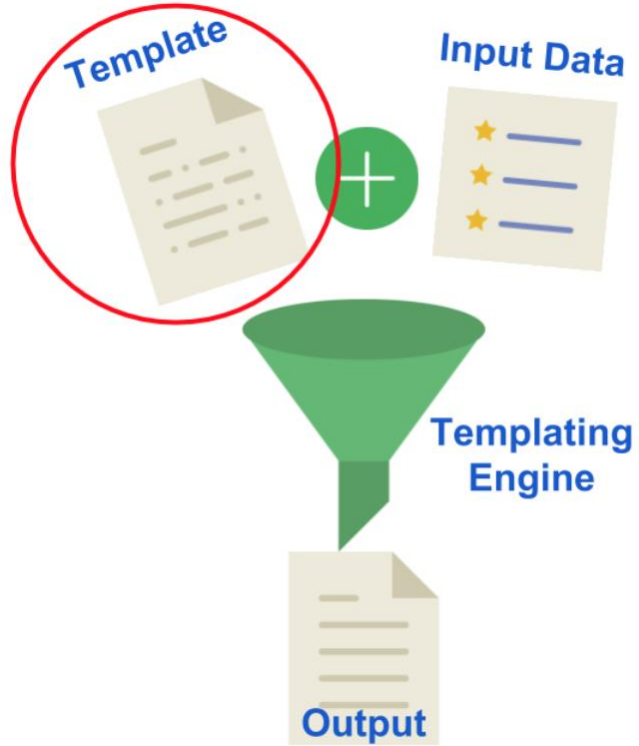


Modelo Vista Controlador

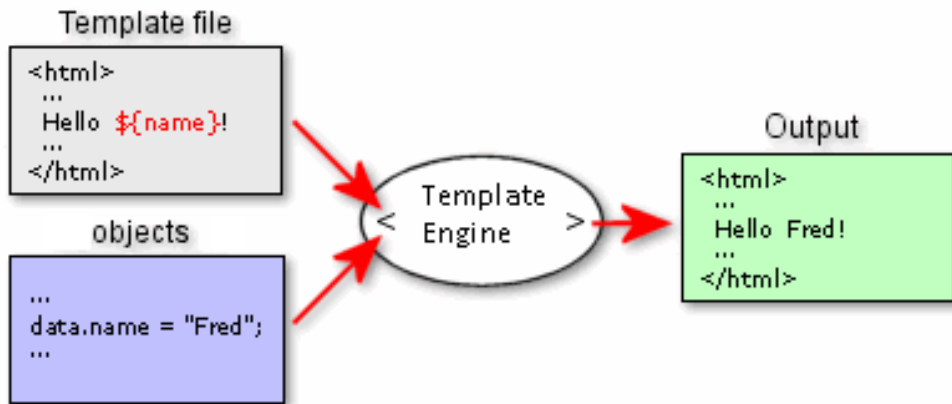
- Un motor de plantillas lee un **archivo de texto** (plantilla) que contiene la presentación ya preparada en un **lenguaje Pseudo HTML** e inserta en él la **información dinámica** que le ordena el "controlador" (la C de MVC) que representa la parte que **une la vista con la información**.
- La sintaxis a utilizar depende del motor de plantillas utilizado.
- Los motores de plantillas suelen tener un pequeño lenguaje de script que permite generar código dinámico.

Motores de Plantillas





Motores de Plantillas



Motores de Plantillas

- Nos permite **separar la lógica** de cliente de la del servidor (frontend y backend), **reduciendo la complejidad** del proyecto final.
- Permite tener **equipos diferentes** trabajando en frontend y backend.
- Permiten desarrollar código **reutilizable**.
- El código es más **fácil de mantener y comprender**.
- Permite agregar **condicionales y estructuras repetitivas** que hacen más **dinámica** la página final con **menos código**.

¿Por qué usar plantillas?

Handlebars



- **Handlebars** es un lenguaje de plantillas simple.
- Utiliza una plantilla y un objeto de entrada para generar HTML u otros formatos de texto.
- Las plantillas de Handlebars tienen el aspecto de texto normal con *expresiones de Handlebars incrustadas*.
- *Una expresión de Handlebars* se compone de `{{ + algunos contenidos + }}`
- Cuando se **ejecuta** la plantilla, las *expresiones de Handlebars* se reemplazan con **valores** de un objeto de entrada.

Handlebars



Ejemplo Handlebars Online

 Handlebars 

Template

```
<p>Hola {{nombre}} {{apellido}}</p>
```

Input

```
{  
  nombre: "Juan",  
  apellido: "Perez",  
}
```

Output

```
<p>Hola Juan Perez</p>
```

<https://handlebarsjs.com/examples/simple->

Vamos al IDE... 



***¿Alguna pregunta hasta
ahora?***



Datos personales

Vamos a practicar lo aprendido hasta ahora

Tiempo: 10 minutos

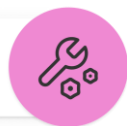


Realizar una página web que permite mostrar datos personales de la siguiente forma:

```
<h1>Datos Personales</h1>
<ul>
  <li>(nombre)</li>
  <li>(apellido)</li>
  <li>(edad)</li>
  <li>(email)</li>
  <li>(teléfono)</li>
</ul>
```

Con los datos que provienen desde un objeto:

```
{
  nombre: '...',
  apellido: '...',
  edad: ...,
  email: '...',
  telefono: '...'
}
```



Importar Handlebars vía CDN en el frontend para crear dicha vista en forma dinámica. Esta página será servida desde el espacio público de un servidor basado en node.js y express.



***¿Alguna pregunta hasta
ahora?***



BREAK

¡10 MINUTOS Y VOLVEMOS!

Motores de plantillas para Express

Fuente: <https://expressjs.com/>

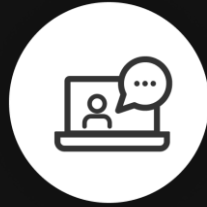
CODER HOUSE

Creando un motor de plantillas custom para express



- Utilizamos el método `app.engine('ext', callback)` para crear nuestro propio motor de plantilla. **ext** hace referencia a la extensión de archivo y **callback** es la función de motor de plantilla, que acepta como parámetros la ubicación del archivo, el objeto options y la función callback.
- El método `app.set('views', path)` especifica la carpeta de plantillas.
- El método `app.set('view engine', 'nombre')` registra el motor de plantillas.
- Luego debemos renderizar desde el objeto respuesta en el endpoint que lo requiera `res.render('plantilla', data)`

Vamos al IDE... 



***¿Alguna pregunta hasta
ahora?***



Motor de plantillas custom

Crearemos nuestro propio motor de plantillas.

Tiempo: 10 minutos

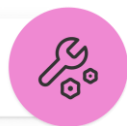


Desarrollar un motor de plantillas custom para un servidor basado en express, que permita representar en la ruta '/cte1' el siguiente archivo de plantilla 'plantilla1.cte':

```
<h1>^^titulo$$</h1>
<p>^^mensaje$$</p>
<b>^^autor$$</b>
<hr>
<i><b>Versión: ^^version$$</b></i>
```

Con los datos que provienen desde un objeto:

```
{
  titulo: (algún título en string),
  mensaje: (algún mensaje en string),
  autor: (algún autor en string),
  version: (numerica)
}
```

Este motor personalizado debe permitir parsear objetos de datos con claves dinámicas y volcar sus valores en la plantilla seleccionada.

Crear otra ruta '/cte2' que represente otro archivo de plantilla: 'plantilla2.cte' con los datos nombre, apellido y la fecha/hora provenientes de un objeto.



***¿Alguna pregunta hasta
ahora?***

Handlebars en express



+

express

+



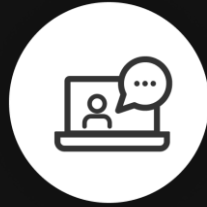
Introducción

- Handlebars puede funcionar de dos formas
 - desde el **lado del servidor**
 - desde el **lado del cliente**.



Esta **versatilidad** hace que podamos decidir mejor cómo queremos realizar nuestras aplicaciones, ya que si es una **SPA** tal vez el enfoque del **lado del cliente** sea más sencillo y útil, pero si queremos un **website** tal vez generar todo en el **servidor** sea más útil.

Vamos al IDE... 



***¿Alguna pregunta hasta
ahora?***



Handlebars con express

Tiempo: 10 minutos



- Transformar el primer desafío, pero esta vez la página dinámica la creará el servidor desde handlebars instalado y configurado para trabajar con express.
- Utilizar la misma estructura de plantilla HTML dentro de una pagina web con encabezado y el mismo objeto de datos.
- El servidor escuchará en el puerto 8080 y el resultado lo ofrecerá en su ruta root.

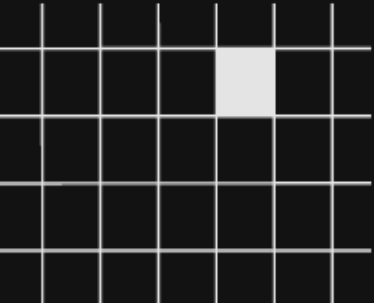
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Introducción a los motores de plantillas y su uso con Express
 - Motor de plantillas Handlebars
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN