

0101010
0100101
1101010

UD10.3- Definición de vistas dinámicas con Thymeleaf

Programación – 1er DAW/DAM

0. CONTENIDOS

- INTRODUCCIÓN
- ¿QUÉ ES THYMELEAF?
 - ARCHIVOS DE VISTA
 - SINTAXIS BÁSICA
 - Textos
 - Acceso a Variables
 - Definición de URL's
- PROCESADO DE UNA PETICIÓN
- CONDICIONALES
- ITERADORES
- WEBGRAFIA

1. INTRODUCCIÓN

- Ya sabemos lo que son los **controladores** y cómo **enviarles información** en forma de parámetros desde un **documento web** para que estos realicen una tarea y nos **respondan** con **otro documento HTML** creado a partir de la información recibida.
 - Mostrar el **detalle de una tarea** a partir de su código.
 - Añadir una tarea nueva al sistema.
 - Borrar una tarea del sistema.

Listado de tareas

[Añadir Tarea](#)

Codigo	Usuario	Descripción	Vencimiento	Acciones	
0	Alex	Hacer la cama	02-02-2022 08:30	Borrar	Detalle
1	Alex	Hacer la comida	08-02-2022 12:30	Borrar	Detalle
2	Pepe	ir a comprar	12-02-2022 17:30	Borrar	Detalle
3	Juan	Estudiar	22-02-2022 08:30	Borrar	Detalle

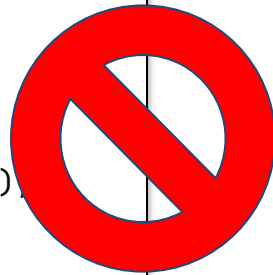
Recomendado para ti en Libros



1. INTRODUCCIÓN

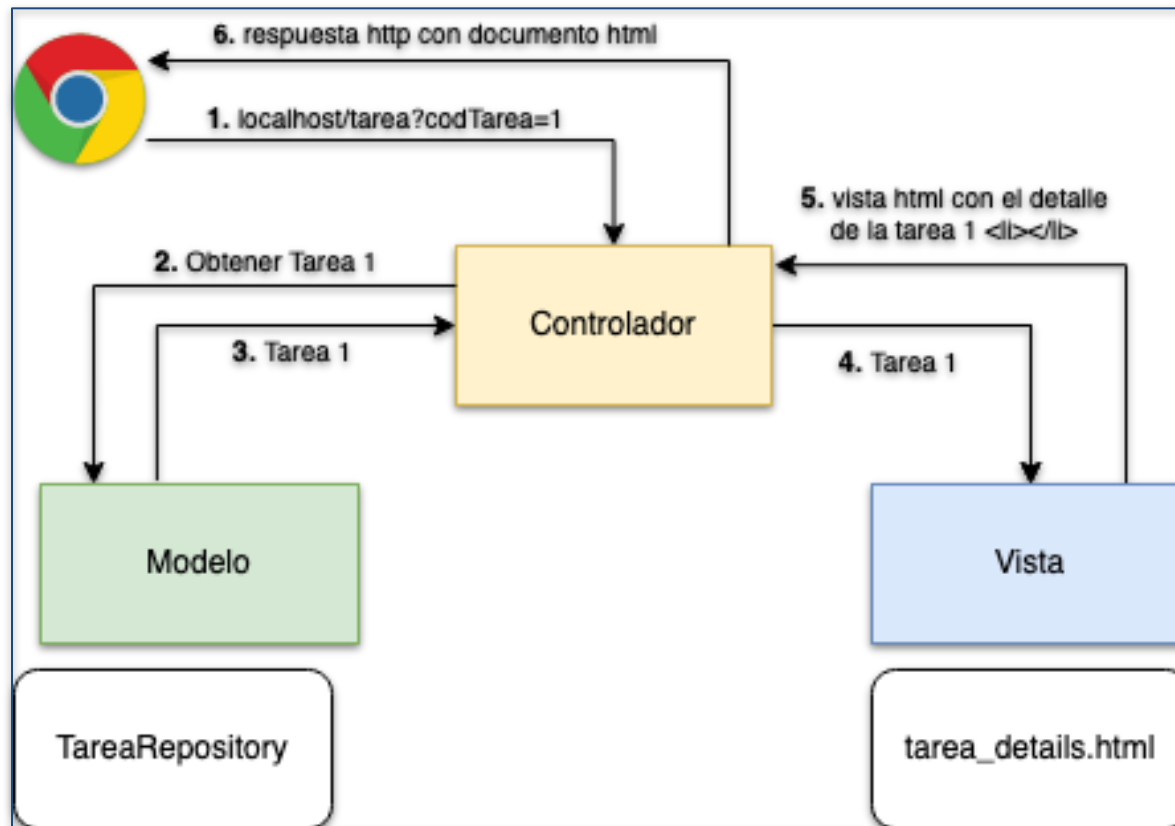
- Pero la estrategia que hemos seguido presenta ciertos **inconvenientes**....
 - **Generar el documento html** desde el controlador:
 - Es engorroso / lioso
 - No seguimos la **arquitectura de capas** estudiada en las unidades anteriores.

```
@GetMapping("/tarea")
public String getTarea(@RequestParam int codTarea) {
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("<html>");
    final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MMMM-yyyy 'a las' hh:mm");
    try {
        Tarea tarea = tareaRepository.get(codTarea);
        stringBuilder.append("<h1>Detalle Tarea ").append(codTarea).append("</h2>");
        stringBuilder.append("<ul>")
            .append("<li><strong>Codigo</strong>: ").append(tarea.getCodigo()).append("</li>")
            .append("<li><strong>Nombre</strong>: ").append(tarea.getUsuario()).append("</li>")
            .append("<li><strong>Descripción</strong>: ")
        stringBuilder.append(tarea.getDescripcion()).append("</li>")
            .append("<li><strong>Vencimiento</strong>: ").append(tarea.getCreadoEn().format(formatter))
            .append("</ul>")
    } catch (NotFoundException e) {
        stringBuilder.append("La tarea con código ").append(codTarea).append(" no existe");
    }
    stringBuilder.append("</html>");
    return stringBuilder.toString();
}
```



1. INTRODUCCIÓN

Arquitectura MVC con Thymeleaf - CONSULTAR DETALLE TAREA



(Entidades,
repositorios...)

Thymeleaf

2. ¿QUÉ ES THYMELEAF?

- **Thymeleaf** es un **motor de plantillas HTML5** del **lado del servidor** para aplicaciones **Java** que permite la **implementación de vistas**
- Las **páginas Thymeleaf** **mezclan código HTML** con una **série de etiquetas** que serán **procesadas** por el motor de plantillas creando **código HTML**.
 - **Convierte la información** recibida de los **controladores en elementos HTML** válidos; *tablas, listados, ...*

Nombre	Email	Password	Fecha de nacimiento
Juan Perez	jperez@daw.es	\$sdakdjsak	10/01/2000
Alex García	alex@daw.es	\$dsakjdskla	10/01/1998
Elena Juarez	elena@daw.es	\$dsadjskadsa	10/01/1998

1. INTRODUCCIÓN

- Por otro lado, si creamos **formularios** como documentos **html estáticos**...
 - No podríamos tener **campos con información dinámica**.
 - Si necesitamos **modificar los datos de los campos**, ¿cómo cambiarlos (para, por ejemplo, precargarlos con información de una fuente de datos)?

Modificar Tarea

Fecha de Entrega Hora de Entrega

Seleccione una categoría ☒ deportes

Seleccione la prioridad ☐ ocio/tiempo libre

☐ jugar al padel

☐ estudiar

☐ tareas del hogar

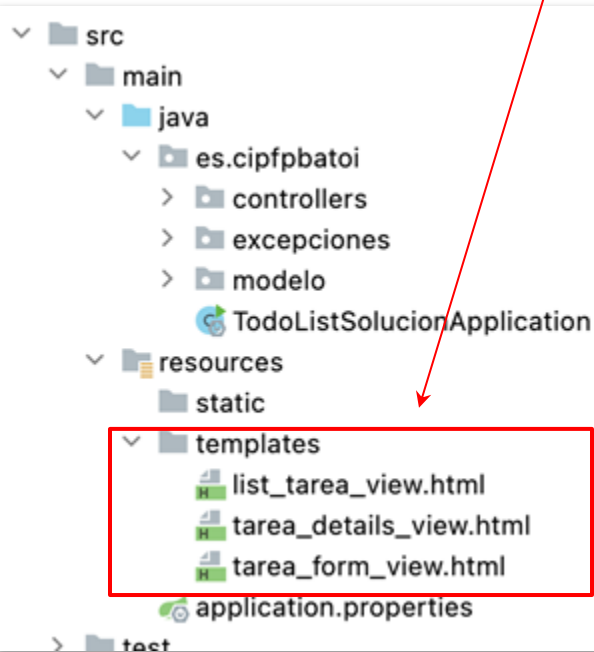
Realizada ☐

Estos **datos** están en la **BD** y necesitamos **pre-cargarlos** al **mostrar el formulario**.

Las **categorías disponibles** están almacenadas en una **tabla/fichero** de la Base de datos

2.1 ARCHIVOS DE VISTA (I)

- Los archivos deben ser extensión **.html**
- Deben estar almacenados en la **carpeta** "resources/templates" del proyecto.
- El prefijo **th:** le indica al motor de plantillas que dicho atributo debe ser **procesado**.



th:text: Establece el **valor text** del elemento **span** al usuario responsable de la tarea

```

<html xmlns:th="http://www.thymeleaf.org" lang="es">
<head>
<title>
<meta charset="UTF-8" />
</head>
<body>
<h1>Detalle de tarea <span th:text="${tarea.getCodigo()}"></span></h1>
<ul>
<li>
<strong>Nombre</strong>:
<span th:text="${tarea.getNombre()}"></span>
</li>
<li>
<strong>Descripción</strong>:
<span th:text="${tarea.getDescripcion()}"></span>
</li>
<li>
<strong>Fecha vencimiento</strong>:
<span th:text="${#temporals.format(tarea.getCreadoEn(), 'dd-MM-yyyy
HH:mm')}"></span>
</li>
</ul>
</body>
</html>
  
```


2.1 ARCHIVOS DE VISTA II

- Nos permite establecer **el valor** de cualquier **atributo** de un **elemento HTML**.

`<etiqueta atributo = "valor" > contenido </etiqueta>`

th:text: Establece el **valor text** del elemento **span** al usuario responsable de la tarea

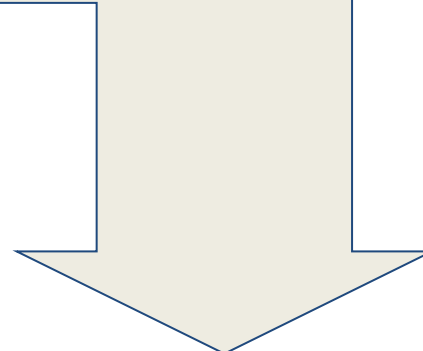
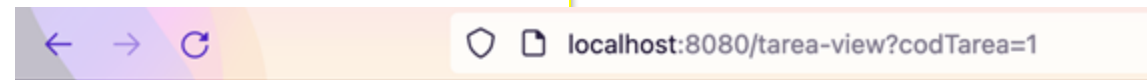
``

El contenido del **atributo text** se **mostrará aquí** cuando sea procesado

2.1 ARCHIVOS DE VISTA III

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<title>Gestor de Tareas</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<body>
<h1>Detalle Tarea <span th:text="${tarea.getCodigo()}"></span></h1>
<ul>
<li>
<strong>Nombre</strong>:
<span th:text="${tarea.getNombre()}"></span>
</li>
<li>
<strong>Descripción</strong>:
<span th:text="${tarea.getDescripcion()}"></span>
</li>
<li>
<strong>Fecha vencimiento</strong>:
<span th:text="${#temporals.format(tarea.getCreadoEn(), 'dd-MM-yyyy
HH:mm')}"></span>
</li>
</ul>
</body>
</html>
```

Todos los **atributos** con **prefijo th:** serán **procesados** por el motor de plantillas para crear un html válido

Detalle Tarea 1

- **Nombre:** Alex
- **Descripción:** Hacer la comida
- **Fecha vencimiento:** 08-02-2022 12:30



2.1 ARCHIVOS DE VISTA IV

- Es importante que veas que, **la información recibida** por el navegador, **NO ES LA PÁGINA THYMELEAF original**, sino el **código html generado** por el motor de plantillas.

Detalle Tarea 1

- Nombre: Alex
- Descripción: Hacer la comida
- Fecha vencimiento: 08-02-2022 12:30

Anterior
Siguiente
Recargar
Añadir página a marcadores...
Guardar como...
Guardar página en Pocket
Seleccionar todo
Hacer captura de pantalla
Ver código fuente de la página
Inspeccionar propiedades de accesibilidad
Inspeccionar

```
1 <!DOCTYPE HTML>
2 <html lang="es">
3 <title>Gestor de Tareas</title>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5 <body>
6 <h1>Detalle Tarea <span>1</span></h1>
7 <ul>
8   <li>
9     <strong>Nombre</strong>: <span>Alex</span>
10  </li>
11  <li>
12    <strong>Descripción</strong>: <span>Hacer la comida</span>
13  </li>
14  <li>
15    <strong>Fecha vencimiento</strong>: <span>08-02-2022 12:30</span>
16  </li>
17 </ul>
18 </body>
19 </html>
20
```

2.2 SINTAXIS BÀSICA. Textos

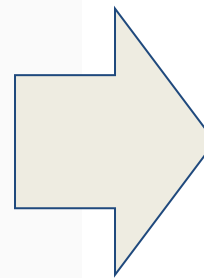
th:text

Permite establecer el atributo texto de cualquier **elemento html**.

```
<span th:text="Hola Mundo"></span>
```

```
<h1 th:text="Hola Mundo 2"></h1>
```

```
<h2 th:text="Hola Mundo 3"></h2>
```



```
<span>Hola Mundo</span>
```

```
<h1>Hola Mundo 2</h1>
```

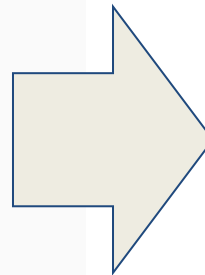
```
<h2>Hola Mundo 3</h2>
```

2.2 SINTAXIS BÀSICA. Textos

th:value

Permite establecer el atributo value de un **elemento input**

```
<input th:value="Hola Mundo">  
<input th:value="Hola Mundo 2">  
<input th:value="Hola Mundo 3">
```



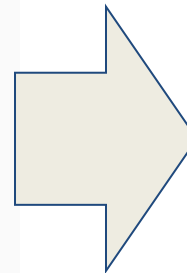
```
<input type="text" value="Hola Mundo">  
<input type="text" value="Hola Mundo 2">  
<input type="text" value="Hola Mundo 3">
```

2.3 SINTAXIS BÀSICA. Acceso a Variables

`${.....}`

Permite realizar **llamadas a métodos** o **acceder al contenido** de una de una variable.

```
<span th:text="${tarea.codigo}"></span>  
<h1 th:text="${tarea.getUsuario()}"></h1>  
<input th:text="${tarea.descripcion}" />
```



```
<span>1</span>  
<h1>Batoi</h1>  
<input>Ir al médico</input>
```

2.3 SINTAXIS BÀSICA. Acceso a Variables

|literal \${nombreVariable} literal| :

Permite **combinar literales** con variables, llevando a cabo la sustitución de los mismos

'texto' + \${nombreVariable}:

Permite **concatenar literales** y variables devolviendo un texto

```
<h2 th:text="|Tarea con código ${tarea.codigo}|"></h2>  
// <h2>Tarea con codigo 1</h2>
```

```
<h2 th:text="'Tarea con código' + ${tarea.codigo}"></h2>  
// <h2>Tarea con codigo 1</h2>
```

2.4 SINTAXIS BÀSICA. Definición de URL's

@{...} :

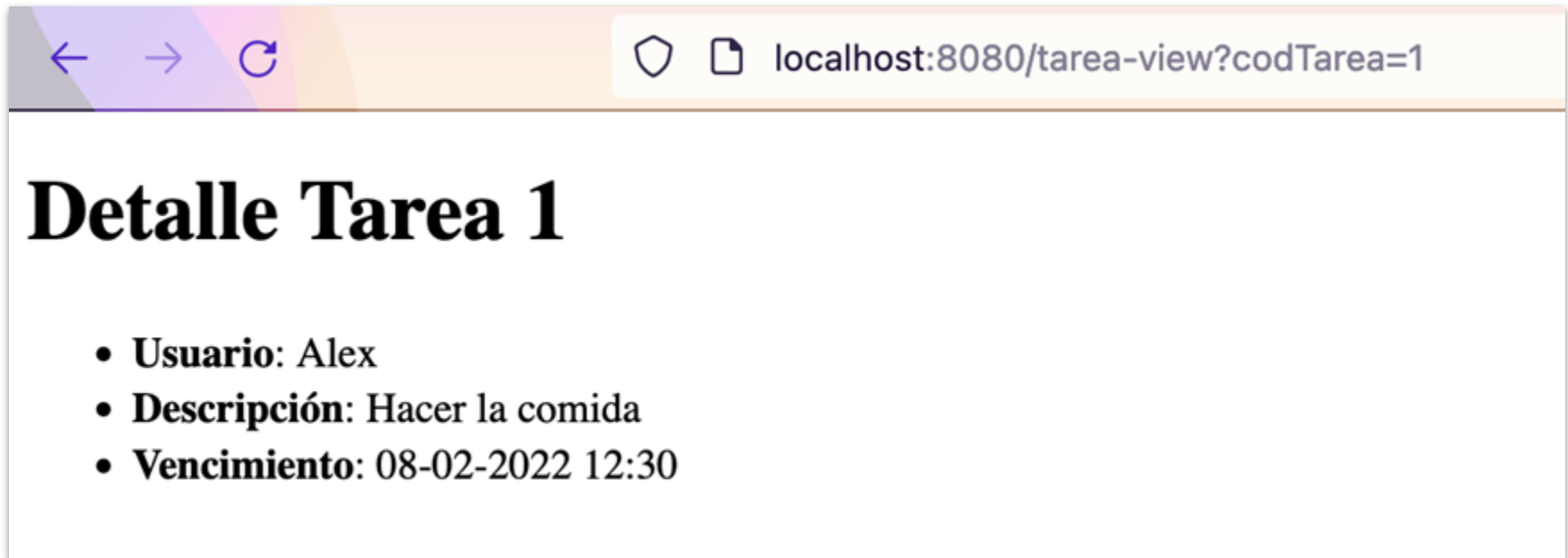
Permite la definición de URL válidas con parámetros. Si se necesita más de un parámetro, estos irán separados por una coma (,)

```
<a th:href="|/tarea?codTarea=${item.codigo}|">Ver Detalle</a>
```

```
<a th:href="@{/tarea(codTarea=${item.codigo})}">Ver Detalle 2</a>
```


3. CREANDO MI PRIMER ENDPOINT. EJEMPLO

- Vamos a ejemplificar como modificar el **endpoint** que nos **permite mostrar el detalle de una tarea** a partir del **código** para que utilice thymeleaf.



3.1 DEFINICIÓN DE DEPENDENCIAS

- Recuerda que en la presentación anterior ya hemos añadido la dependencia *Thymeleaf*. Debemos asegurarnos de tenerla para poder **hacer uso** de toda la **potencia de este sistema de plantillas**. **Compruebalo mirando el fichero *pom.xml* del proyecto.**

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

3.2 CREAMOS EL ARCHIVO DE VISTA

- Creamos un **archivo .html** dentro de la carpeta `/resources/templates` llamado `tarea_details_view.html` con el html que queremos conseguir

`/templates/tarea_details_view.html`

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<body>
<div>
  <h2>Detalle de la Tarea <span>Codigo de la Tarea</span></h2>
  <ul>
    <li>
      <strong>Usuario: </strong>
      <span>Datos Usuario</span>
    </li>
    <li>
      <strong>Descripción: </strong>
      <span>Descripción de la tarea a realizar</span>
    </li>
    <li>
      <strong>Vencimiento </strong>
      <span>Vencimiento</span>
    </li>
  </ul>
</div>
</body>
</html>
```

Todos los **datos remarcados** son los que **queremos que sean sustituidos** por los **datos de la tarea** que a **visualizar**

3.3 DEFINICIÓN DEL CONTROLADOR

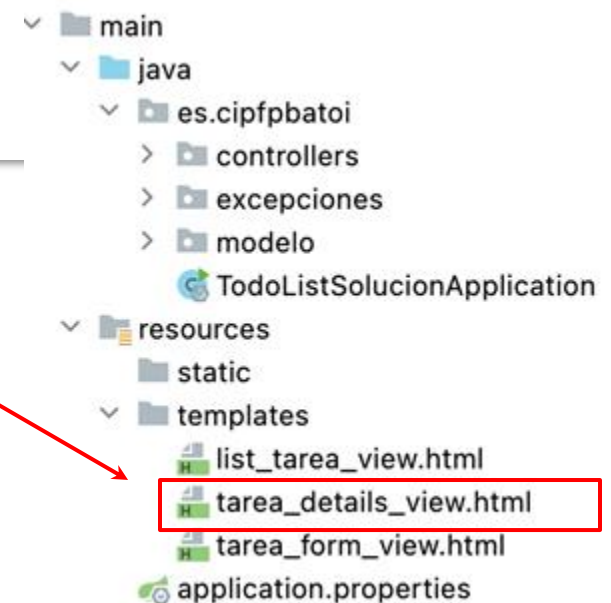
Cargando la Vista

- Definimos el **nuevo controlador** que se encargará de **cargar la vista** que hemos creado en el **paso anterior**.

```
@GetMapping("/tarea")  
public String example4Action() {  
    return "tarea_details_view";  
}
```

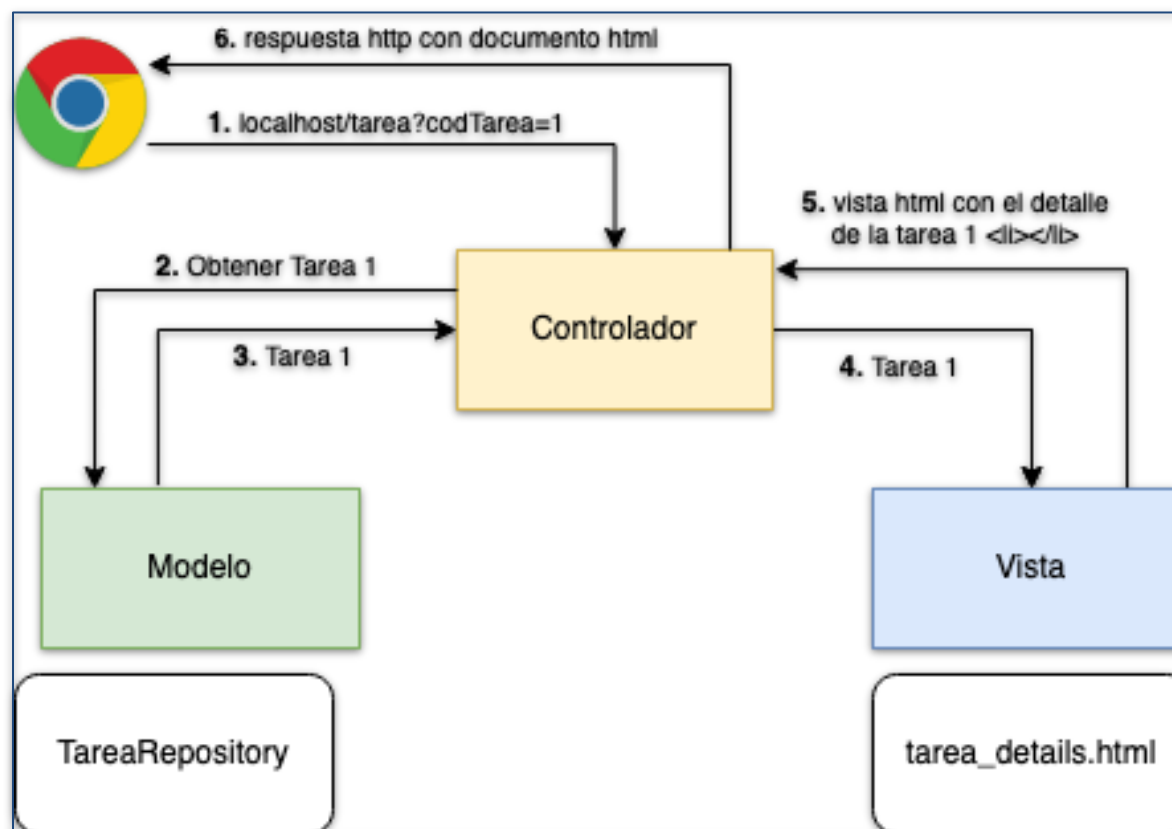
NO hemos incluido la anotación
`@ResponseBody`

Retornamos el nombre del **archivo.html** que **contiene la vista** a **cargar** (la vista **debe estar** dentro de la **carpeta templates**)



3.4 PASO DE PARÁMETROS A LA VISTA

- Cuando el **controlador**, carga **la vista asociada** a la petición, deberá **pasarle** los parámetros **que necesita**. Por ejemplo, **la tarea** de la que se **quiere consultar información**
 - **Estos datos** son **obtenidos**, generalmente, de la **capa de persistencia**.



3.4 PASO DE PARÁMETROS A LA VISTA II

- Debemos **indicarle al framework** que queremos acceder al **modelo de la vista** para **establecer los datos** que queremos que la vista tenga accesibles.
 - Spring Boot, inyectará un objeto de tipo **Model** al cual debemos acceder y **añadir los objetos** (DTO) con la información a la que **necesitamos acceder desde la vista**.

```
@GetMapping("/tarea")  
public String example4Action(@RequestParam String codTarea,  
                             Model model) {  
    Tarea tareaBuscada = tareasRepository.get(codTarea);  
    model.addAttribute("tarea", tareaBuscada);  
    return "tarea_details_view";  
}
```

En la **vista** podremos **acceder** a todos sus **métodos**
`${tarea.xxxx()}`

3.5 ACCESO A DATOS DESDE LA VISTA

Acceso a datos desde la Vista

- Podremos **acceder** al **objeto 'tarea'**, haciendo uso de las **sentencias vistas anteriormente**.

/templates/tarea_details_view.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<body>
<div>
  <h2 th:text="'Tarea con código' + ${tarea.codigo}"></h2>
  <ul>
    <li>
      <strong>Usuario: </strong>
      <span th:text="${tarea.usuario}"></span>
    </li>
    <li>
      <strong>Descripción: </strong>
      <span th:text="${tarea.descripcion}"></span>
    </li>
    <li>
      <strong>Vencimiento: </strong>
      <span th:text="${#temporals.format(tarea.vencimiento, 'dd-MM-yyyy HH:mm')}"></span>
    </li>
  </ul>
</div>
</body>
</html>
```

Si especificamos un **atributo**, el motor de plantillas **llamará automáticamente a su getter** para obtener el dato

ACTIVIDAD PREVIA

Actividad 5.- Modifica el controlador que muestra el detalle de una tarea a partir de su código para que muestre los datos utilizando el motor de **plantillas thymeleaf**.

Detalle Tarea 2

- **Código:** 2
- **Nombre:** Pepe
- **Descripción:** ir a comprar
- **Vencimiento:** 12-febrero-2022 a las 05:30
- **Realizado:** No
- **Prioridad:** MEDIA

Consulta realizada por el usuario **Admin**

Información obtenida de un **objeto tarea**

Información obtenida de una **variable de tipo String** llamada **agente** que pasaremos a la vista y que **contendrá el nombre del alumno**

url: /tarea
query params: codTarea
nombre vista: tarea_details_view.html
datos pasado a la Vista:
tarea: Tarea
Agente: String

4. CONDICIONALES

Simple

th:if="{condicion}": permite mostrar un elemento si SE CUMPLE la condición.

th:unless="{...}": permite mostrar un elemento si NO SE CUMPLE la condición.

```
<!-- IF. Ejemplo: suponemos que sólo queremos mostrar la tarea si su código es mayor que cero -->
<div th:if="{tarea.codigo} > 0">    También sería válido hacer: <div th:if="{tarea.codigo > 0}">
    <p><strong>Usuario: </strong><span th:text="{tarea.usuario}"></span></p>
    <p><strong>Descripción: </strong><span th:text="{tarea.descripcion}"></span></p>
    <p><strong>Vencimiento: </strong>
        <span th:text="{#temporals.format(tarea.vencimiento, 'dd-MM-yyyy HH:mm')}"></span>
    </p>
</div>
<!-- ELSE -->
<div th:unless="{tarea.codigo} > 0">
    <p>Tarea con código no válido</p>
</div>
```

4. CONDICIONALES II

Unarios

`th:...="${condicion} ? 'valor positiu' : 'valor negatiu' "`:

Permiten **mostrar un elemento u otro** en función de la evaluación de **una condición**.

```
<span th:text="${tarea.realizada} == true? 'SI' : 'NO'"></span>
```

...

```
<tr th:class="${row.num % 2 == 0}? 'even' : 'odd'">... </tr>
```

Establecemos el **atributo class** del elemento **tr** a “**odd**” o “**even**” en función de si el **número de fila** es **par o impar**

5. ITERADORES

th:each Permite recorrer **listados** y **colecciones** con el fin de crear **elementos html** para cada uno de los items que contiene. En el listado de tareas, cada una de las tareas debe convertirse en un **<tr>** de elemento **table**.

```
<table>
  <tr>
    <th>Código</th>
    <th>Usuario</th>
    <th>Descripción</th>
    <th>Vencimiento</th>
    <th>Acciones</th>
  </tr>
  <tr th:each="item : ${tareas}">
    <td th:text="${item.codigo}"></td>
    <td th:text="${item.usuario}"></td>
    <td th:text="${item.descripcion}"></td>
    <td th:text="${#temporals.format(item.vencimiento, 'dd-MM-yyyy HH:mm')}"></td>
    <td><a th:href="@{/tarea-delete(codTarea=${item.codigo})}">Borrar</a></td>
  </tr>
</table>
```

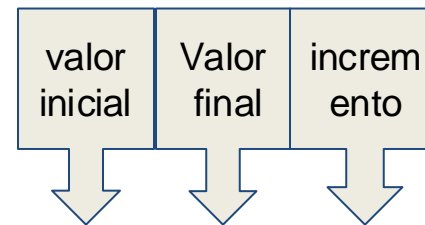
Listado de tareas

Codigo	Usuario	Descripción	Vencimiento	Acciones
0	Alex	Hacer la cama	02-02-2022 08:30	Borrar
1	Alex	Hacer la comida	08-02-2022 12:30	Borrar
2	Pepe	ir a comprar	12-02-2022 17:30	Borrar
3	Juan	Estudiar	22-02-2022 08:30	Borrar

5.1 ESTRUCTURA for

Podemos simular un **for** haciendo **uso** de **expresiones del lenguaje**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <div th:each="i : ${#numbers.sequence( 1, 4, 1)}">
    <p th:text="${i}"></p>
  </div>
</body>
</html>
```

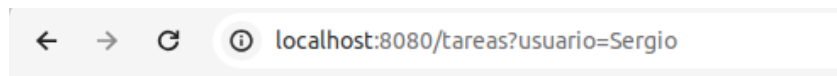


```
1
2
3
4
```

ACTIVIDAD PREVIA

Actividad 6.- Crea un nuevo método en el controlador TareaController que nos permita visualizar las tareas pendientes. Recibirá como **parámetro (opcional)** un nombre de usuario:

- A.** Si este **parámetro existe**, se visualizarán sólo las tareas cuyo nombre de usuario coincida con el parámetro recibido. El filtrado debe hacerlo un método del repositorio `public ArrayList<Tarea> findAll(String user)`.
- B.** Si el **parámetro no existe** (no se proporciona) o se proporciona pero **está vacío**, se visualizarán todas las tareas.
- C.** Si **no se encuentran tareas** se visualizará un mensaje descriptivo.

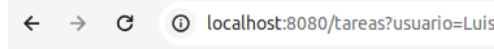


Listado de tareas Sergio

A

[Añadir Ta](#)

Código	Usuario	Descripción	Vencimiento	Acciones
1	Sergio	Hacer la comida	08-02-2024 12:30	Borrar
3	Sergio	Estudiar	22-02-2024 08:30	Borrar

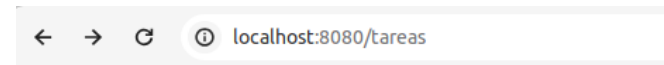


Listado de tareas Luis

[Añadir Tarea](#)

No existen Tareas

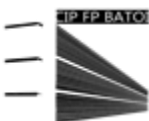
C



Listado de tareas

B

Código	Usuario	Descripción	Vencimiento	Acciones
1	Sergio	Hacer la comida	08-02-2024 12:30	Borrar
2	Raúl	ir a comprar	12-02-2024 17:30	Borrar
3	Sergio	Estudiar	22-02-2024 08:30	Borrar



6. FORMULARIOS CON REDIRECCIÓN

- En ocasiones, **cuando** llevamos a cabo una **inserción de datos** a través de un formulario necesitamos que, **en lugar** de visualizar un **mensaje de confirmación**, **nos muestre otra página**.
 - Por **ejemplo**, queremos que, **al insertar una tarea**, nos **muestre el listado de tareas completo** con la nueva tarea añadida.
 - El proceso por el cual **desde una acción del controlador** nos permite **cargar otra página** se **conoce** como **REDIRECCIÓN**.

Redirección 302

Codigo	Usuario	Descripción	Vencimiento
0	Alex	Hacer la cama	02-02-2022 08:30
1	Alex	Hacer la comida	08-02-2022 12:30
2	Pepe	ir a comprar	12-02-2022 17:30
3	Juan	Estudiar	22-02-2022 08:30
1	Alex	Ir al médico	11-12-2023 12:00
5	Alex	Hacer la cama	02-02-2022 08:30
6	Alex	Hacer la comida	08-02-2022 12:30
7	Pepe	ir a comprar	12-02-2022 17:30
8	Juan	Estudiar	22-02-2022 08:30
10	Alex	Ir al médico - Ejemplo	04-03-2023 12:00

6.1 EJEMPLO I. Paso 4

- Sin **redirección** tenemos....

← → ↻ localhost:8080/tarea-form

Añadir Nueva Tarea

Codigo: 10

Nombre: Alex

Descripción: Ir al médico - Ejemplo

Prioridad: ALTA

realizada: ☒

Fecha vencimiento: 04 / 03 / 2023

Hora vencimiento: 12 : 00

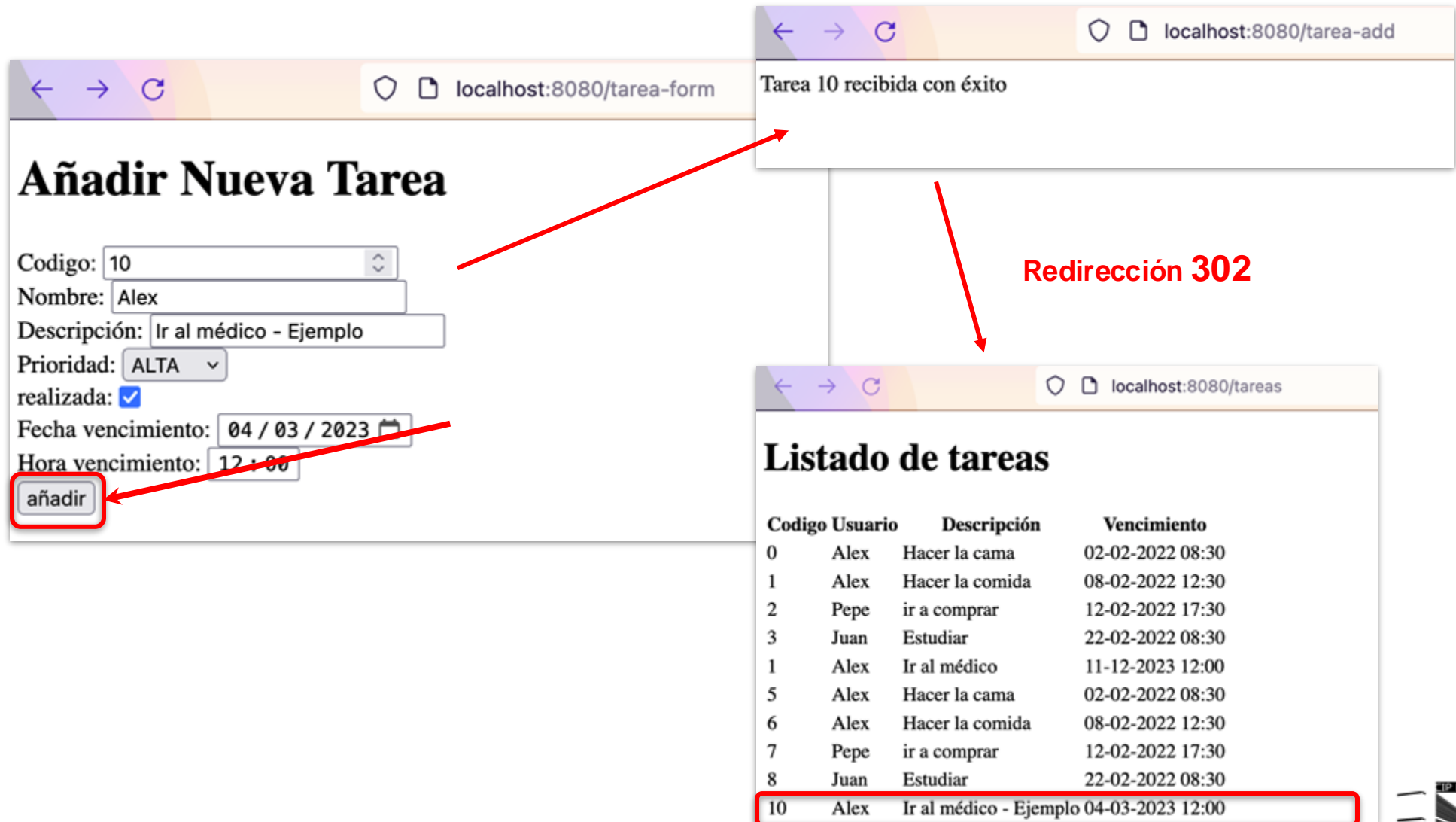
añadir

← → ↻ localhost:8080/tarea-add

Tarea 10 recibida con éxito

6.1 EJEMPLO I. Paso 4 con redirección

- Con redirección



6.1 EJEMPLO I. Paso 4 con redirección

- Para ello, devolveremos **la ruta del nuevo controlador** al que queremos que **acceda el navegador** cuando termine la ejecución.

```
@PostMapping(value = "/tarea-add")
```

```
public String postAddAction(@RequestParam Map<String, String> params) {
```

```
    int code = Integer.parseInt(params.get("code"));
```

```
    String user = params.get("user");
```

```
    String descripcion = params.get("description");
```

```
    String prioridad = params.get("priority");
```

```
    ...
```

```
    Tarea tarea = new Tarea(code, user, descripcion, prioridad, fechaHora, false);
```

```
    tareaRepository.add(tarea);
```

```
    return "redirect:/tareass";
```

```
}
```

Indicamos al navegador que haga una petición GET al controlador /tareass

ACTIVIDAD PREVIA

Actividad 7.- Refactoriza la aplicación de tareas de forma que:

- Cuando se **inserte o borre una tarea** lleve a cabo una **redirección** para mostrar el **listado de tareas**.
- Añade un enlace a **cada fila del listado**, de forma que cuando pulsemos sobre él, **nos lleve al detalle de la tarea**.
- Añade un **enlace en el detalle de la tarea** que al pulsarlo nos **lleve al listado anterior**.
- Añade un enlace al listado para poder añadir **nuevas tareas**.

The image displays three screenshots of a web application interface, connected by red arrows indicating the navigation flow:

- Listado de tareas:** A table showing a list of tasks. The 'Acciones' column contains links for 'Borrar' and 'Detalle'. The 'Añadir Tarea' link is highlighted with a red box.
- Detalle Tarea 3:** A page showing the details of a specific task (Codigo: 3). It includes fields for Nombre, Descripción, Prioridad, Fecha vencimiento, and Hora vencimiento. A 'Volver al listado' link is highlighted with a red box.
- Añadir Nueva Tarea:** A form for adding a new task, with fields for Codigo, Nombre, Descripción, Prioridad, Fecha vencimiento, and Hora vencimiento. An 'añadir' button is at the bottom.

Red arrows indicate the navigation flow: from 'Añadir Tarea' in the 'Listado de tareas' to the 'Añadir Nueva Tarea' form, from 'Detalle' in the 'Listado de tareas' to 'Detalle Tarea 3', and from 'Volver al listado' in 'Detalle Tarea 3' back to the 'Listado de tareas'.

7. WEBGRAFIA

- Documentación Oficial Framework.
<https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html>. **Thymeleaf Team**
- Trabajando con enums en thymeleaf <https://www.baeldung.com/thymeleaf-enums>. **Amy the Gregorio**
- Trabajando con fechas en thymeleaf. <https://www.baeldung.com/dates-in-thymeleaf>. **Baeldung**

- Eso es todo... de momento :-)