Tema 4 – Spring MVC + Thymeleaf

Grado en Ingeniería Informática en Tecnologías de la Información

Departamento de Ingeniería de Sistemas Informáticos y Telemáticos

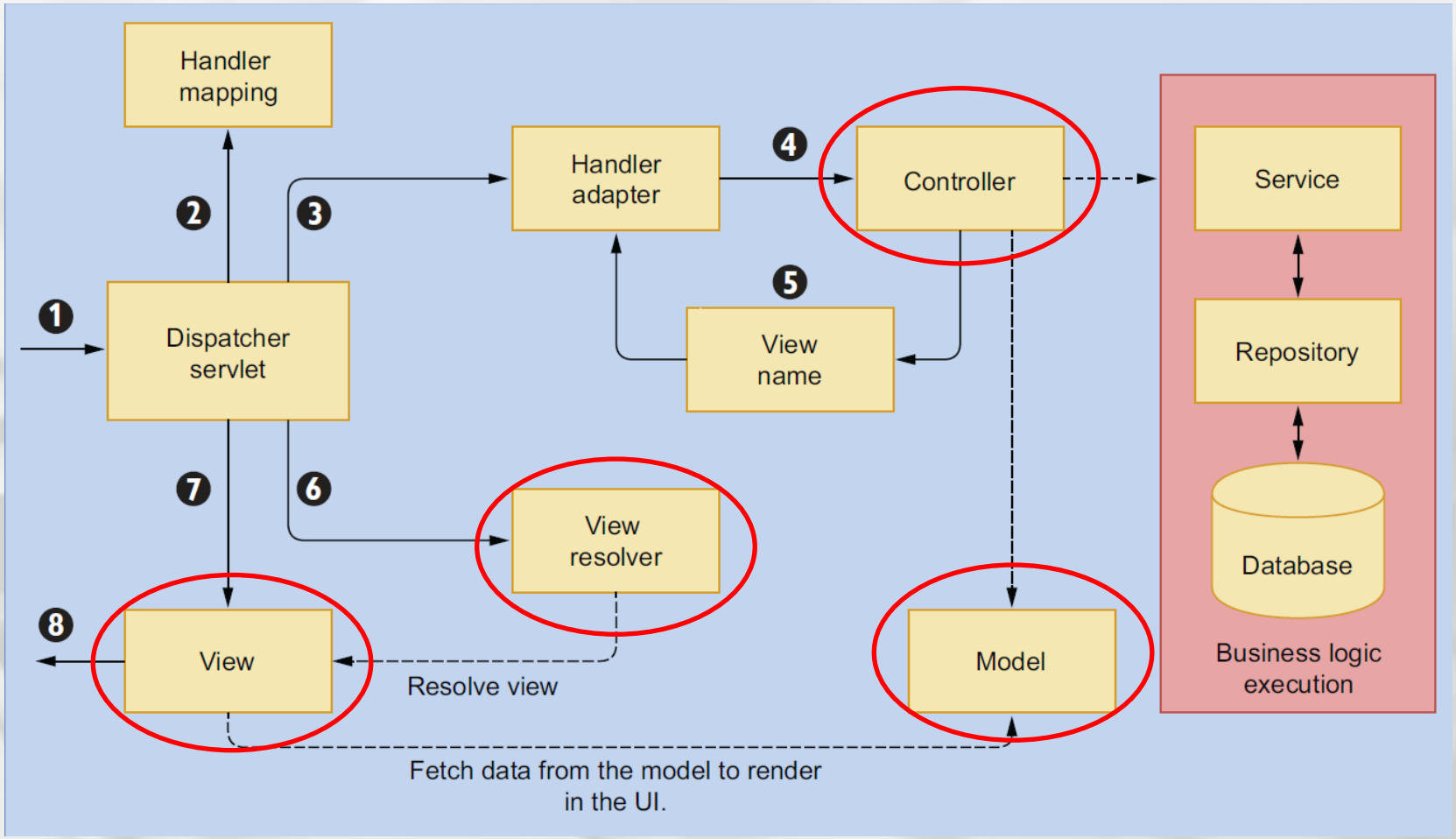
Área de Lenguajes y Sistemas Informáticos

Dr. Luis V. Calderita

Introducción

- **Thymeleaf** es un motor de plantillas del lado del servidor que permite definir varios tipos de plantillas: `HTML`, `XML`, `TEXT`, `JAVASCRIPT`, `CSS y RAW`
- Nos centramos en plantillas HTML.
  - Es una página HTML que contiene etiquetas HTML junto con etiquetas especiales de Thymeleaf.
  - Las etiquetas son procesadas en tiempo de ejecución y son reemplazadas por los datos suministrados.
  - El contenido se muestra en el navegador como HTML plano

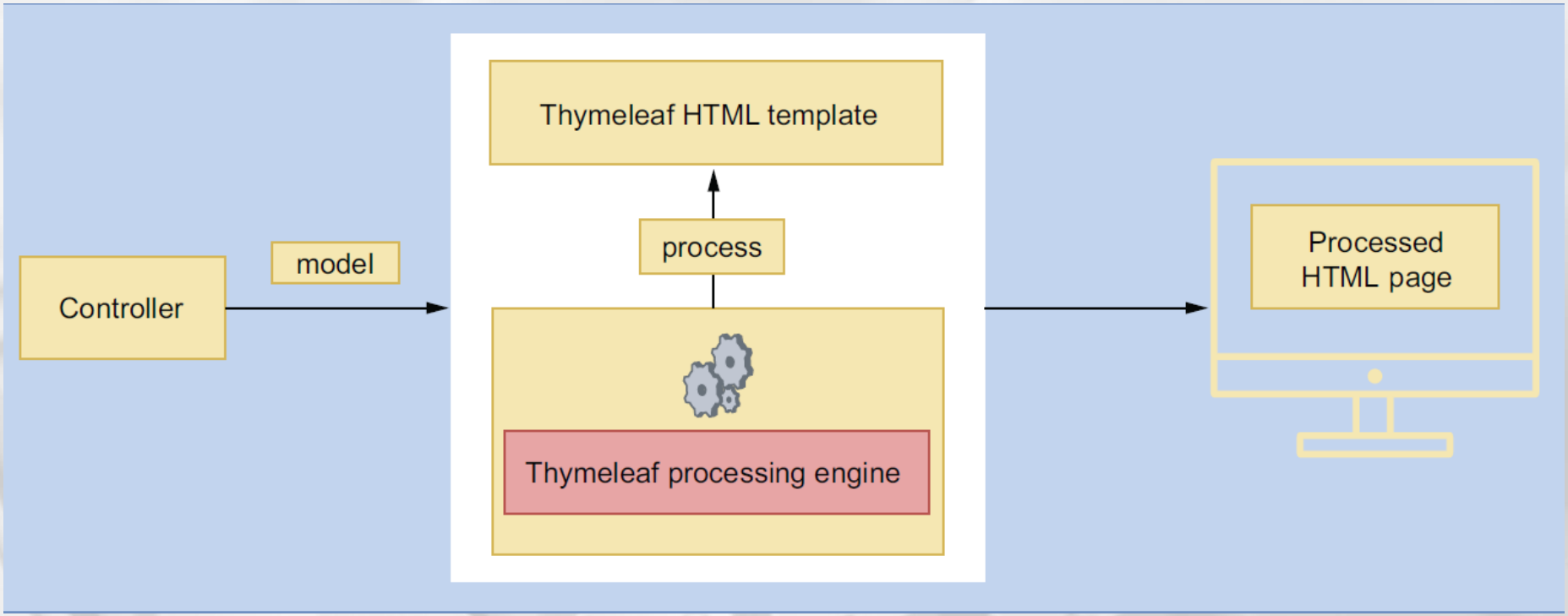
Componentes de Spring MVC



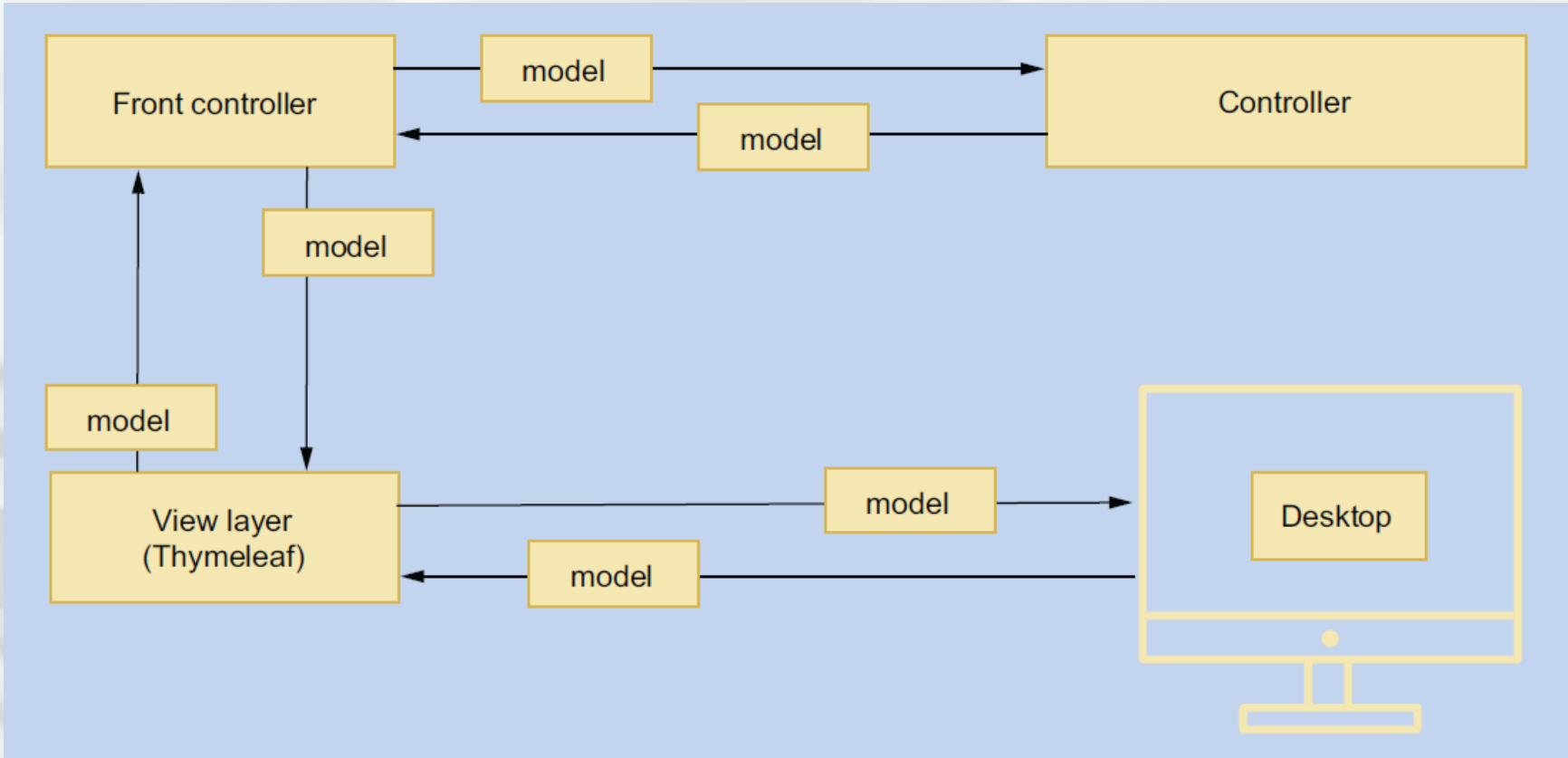
Funcionamiento General

- Se envían datos desde un controlador a través de una estructura de datos (un mapa) a la capa de vista para renderizar los datos en la UI
- A esta estructura de datos se le llama: **model**
- Se envía la vista en la que se renderizan los datos del modelo asociado
- El nombre de la vista se mapea con la página HTML apropiada
- El motor de Thymeleaf procesa las etiquetas específicas para sustituirlas por los datos de la aplicación contenidos en el modelo

Funcionamiento General: Esquema



Patrón de diseño de aplicaciones web con Spring MVC y Thymeleaf



Spring MVC y Thymeleaf.

- En Spring MVC, los controladores preparan un modelo con los datos y proporcionan la vista donde serán renderizados
- **El modelo es un map**, donde la clave es un String (será el nombre de la variable) y el valor un Object de Java
- Este *model map* se transforma en un *objeto/variable de contexto* en Thymeleaf
- El cuál pone todas las *variables* contenidas en el modelo a disposición de la plantilla html.
- Las variables pueden ser accedidas mediante una serie de etiquetas definidas por Thymeleaf

Spring model attributres

- En Spring MVC, los datos que pueden ser accedidos en tiempo de ejecución se conocen como *model attributes*
- El término equivalente en Thymeleaf es *context variables*
- P. Ej. la sintaxis de acceso en Thymeleaf es:
  - ${nombre de la variable}
- Esta sintaxis está escrita en lo que se conoce como Standard Dialect de Thymeleaf

Thymeleaf: Standard Dialects

- *The Standard dialects* (son dos, aunque muy similares) llamados *Standar y **SpringStandard**.*
  - Spring debido a que contiene características para la integración con Spring
  - Proporciona suficientes características para la mayoría de escenarios
  - Se identifican dentro de un template (html) por usar el prefijo th, p. ej.
    -
  - Define una sintaxis para evaluar expresiones (Standard Expression syntax)

Standard Expression Syntax

- Los valores de los atributos pueden ser accedidos mediante las Standard Expression
- Existen cinco tipos:
  - **${...} : Variable expressions.**
  - ***{...} : Selection expressions.**
  - #{...} : Message (i18n) expressions.
    - Para internacionalización. No lo vemos aquí
  - **@{...} : Link (URL) expressions.**
  - ~{...} : Fragment expressions.
    - Para compartir partes comunes (menús, footer…). No las vemos aquí, pero útiles. Más info

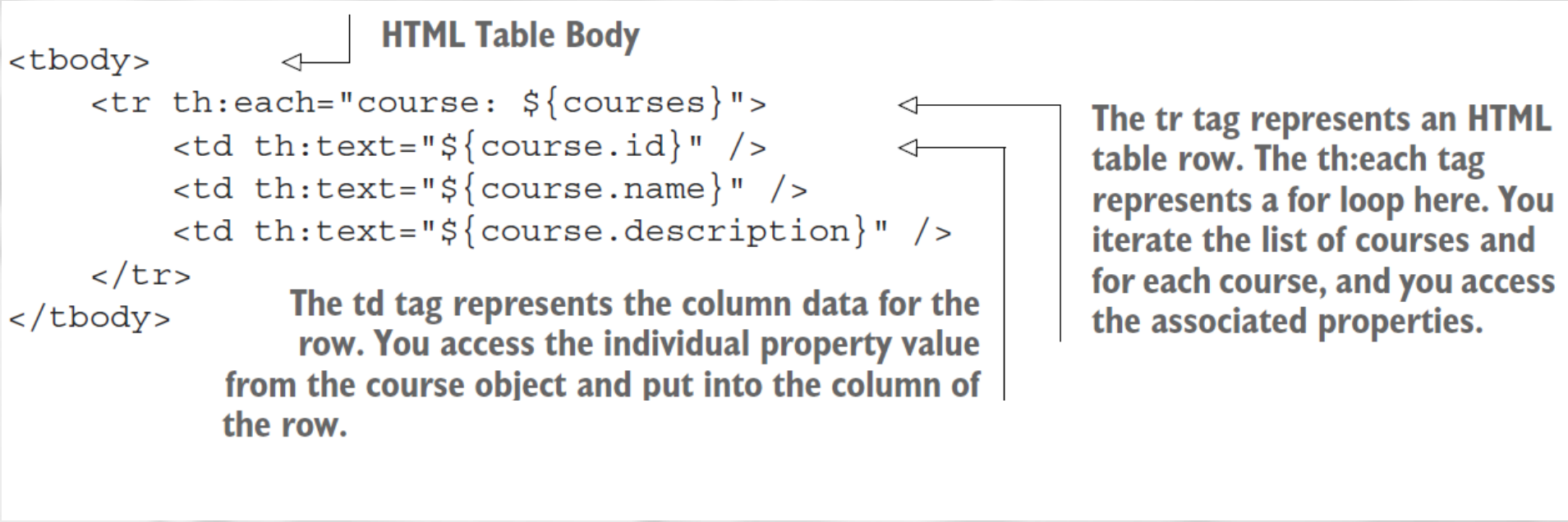
Variable expressions

```
<span th:text="${book.author.name}">
```

- ${…} para acceder a la variable. Van entre “”
- Accedemos a una variable o atributo del modelo llamada *book*
  - **book**, es un objeto, que está compuesto por otro objeto **author**, que a su vez tiene la propiedad **name**
  - Similar a: book.getAuthor().getName()

Variable expressions: th:each

- La variable expressions, son útiles en escenarios más complejos, como bucles, condicionales…



Variable expressions: th:if y th:unless

```
<td>
    <span th:if="${author.gender} == 'M'" th:text="Male" />
    <span th:unless="${author.gender} == 'M'" th:text="Female" />
</td>
```

- Permite renderizar una vista basada en una condición
  - El objeto author, tiene una propiedad gender.
    - Los valores para esta propiedad son M o F
  - El código anterior, permite mostrar en la vista Male o Female, según el caso.

Variable expressions: th:switch y th:case

```
<div th:switch="${course.category}">
    <div th:case="'Spring'">
        <h2>Spring Course</h2>
    </div>
    <div th:case="'JavScript'">
        <h2>JavaScript Course</h2>
    </div>
    <div th:case="*">
        <h2>Some other course:</h2>
    </div>
</div>
```

Using Thymeleaf switch-case statements to evaluate condition

- th:switch y th:case se utilizan para renderizar contenido condicional
  - El caso por defecto se escribe con “*”

Selection expressions

```
<div th:object="${book}">
  ...
  <span th:text="*{title}">...</span>
  ...
</div>
```

- Similares a las variables de expresión, se ejecutan sobre un objeto seleccionado previamente. En lugar de sobre todas las variables/atributos del modelo/contexto
  - th:object permite seleccionar el objeto previamente
  - “*{title}”, * representa el objeto y title una propiedad

Link (URL) expressions

- Usan el tag th:href y @{…}
  - Absolutas:

```
<a th:href="@{http://www.thymeleaf/documentation.html}">
```

- Relativas al contexto: Empiezan con /. Genera la URL: **http://localhost:8080/order/list**

```
<a th:href="@{/order/list}">
```

  - Relativas a la ubicación actual:

Si estás en: **http://mdai.com/miapp/actual**

La URL sería: **http://mdai.com/miapp/actual/pagina**

```
<a th:href="@{pagina}"></a>
```

Link (URL) expressions

- Se pueden usar en todas partes, por ejemplo, en formularios:

```
<form action="#" th:action="@{/addUsuario}"
      th:object="${usuario}" method="post">
```

- Se pueden añadir parámetros:

```
<td><a th:href="@{/updateUsuario/{id}(id=${usuario.id})}"></a></td>
```

Lecturas Recomendadas

- La documentación de Thymeleaf está un poco desordenada…
  - Standard Dialect (recomendado)
    - Getting started with the Standard dialects in 5 minutes
- Como referencia usar estos dos tutoriales:
  - Thymeleaf
  - Thymeleaf + Spring

PRACTICANDO CON THYMELEAF

Añadir Thymeleaf al proyecto I

- A partir del ejemplo en el que incluimos un controlador
- Añadir spring-boot-starter-thymeleaf al POM:

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

- **Nota**: Puedes crear el proyecto Spring Boot de nuevo con la dependencia Thymeleaf y las anteriores (Spring web, Spring Data Jpa, H2 database). Con esto sería suficiente para crear la app-web definitiva

Añadir Thymeleaf al proyecto II

- Mover el fichero myIndex.html de Static a Templates
  - Ahora Thymeleaf se va a encargar de resolver los nombres y proporcionar las plantillas
- Renombrar myIndex.html al común index.html
- Actualizar adecuadamente el controlador “/”:

```
@GetMapping("/")
public String index() {
    System.out.println("\t Recogo la peticion. "
            + "Devuelvo la vista index; "
            + "index.html esta ubicado en Templates");
    return "index";
}
```

Hola Mundo con Thymeleaf

```java
import org.springframework.ui.Model;
@Controller
public class HomeController {

    @GetMapping("/holaTh")
    public String holaPage (Model model) {
        String texto = "Hola mundo con Thymeleaf + Spring";
        model.addAttribute("Bienvenida",texto);
        return "holaTh";
    }
```

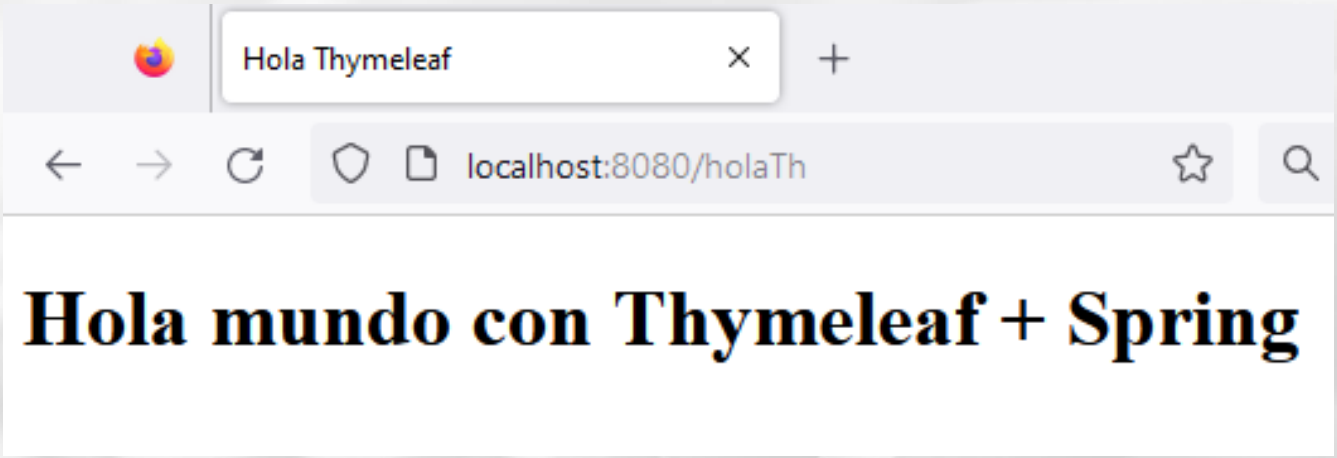
- @GetMapping (“/holaTh”)
  - Recibimos las peticiones a localhost:8080/holaTh
- model.addAttribute (“Bienvenida”,texto)
  - Añadimos al modelo la variable Bienvenida, para que pueda ser procesada en la vista devuelta *holaTh.html*

Código en la vista holaTh.html

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head> <meta charset="utf-8">   <title>Hola Thymeleaf</title>
4  </head>
5  <body>
6
7   <h1 th:text="${Bienvenida}">
8      Aquí iría un texto de Bienvenida. Será reemplazado
9      por el contenido de $Bienvenida en tiempo de ejecución
10  </h1>
11
12 </body>
13 </html>
```

- **holaTh.html**, debe estar en ***resources/templates***
- **Línea 2**: Incluimos el espacio de nombres de Thymeleaf en un .html convencional
- **Línea 7**: Accedemos a la variable o atributo contenida en el modelo, Bienvenida.
  - Usamos ${…} para acceder al contenido
  - Usamos th:text para convertirlo a texto

Vista holaTh.html renderizada



- El código del html ha sido reemplazado por el contenido de la variable $Bienvenida en tiempo de ejecución
- Las variables del modelo están accesibles directamente por su clave

Desarrollando una app-web básica

- Objetivo:
  - Extender el ejemplo de usuario y direcciones para realizar el CRUD desde la vista
- Requisitos
  - Tener implementado hasta el ejemplo anterior con Thymeleaf.
- Pasos (aproximados):
  - Acceder y listar los usuarios
  - Crear un formulario e insertar un usuario nuevo
  - Actualizar un usuario
  - Borrar un usuario

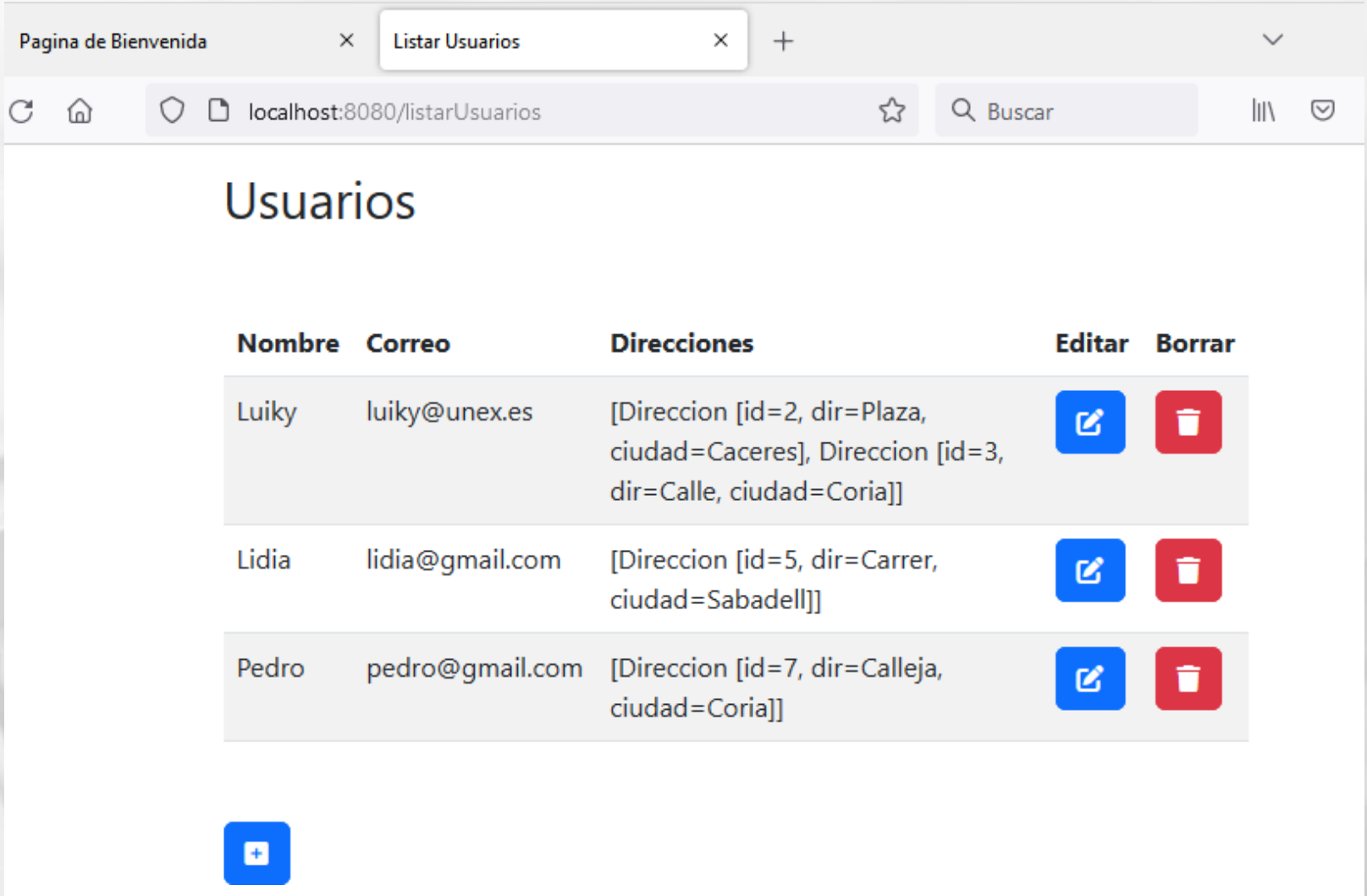
Acceder y listar usuarios I

- Crear un html en resources/templates para mostrar los usuarios del sistema.
  - P. ej. listarUsuarios.html
  - Incluir en él, el espacio de nombres de Thymeleaf
  - *Mostrar sólo un texto de momento…*
- Editar el index.html y añadir un enlace a listarUsuarios.html
- Crear un @Controller que responda a las peticiones relacionadas con los usuarios.
  - Crear un método que responda a la petición: listarUsuarios
- **Ejecutar la app-web y comprobar que funciona**

Acceder y listar usuarios II

- En el método del @Controller invocar al servicio necesario para recuperar los usuarios
- Estudiar Thymeleaf y crear una tabla que muestre todos los usuarios.
  - Mostrar las direcciones como un campo más, no como una tabla
  - Recomendación: Estudiar BootStrap y Fontawesome para que se vea más bonito
- Ejecutar la app-web y comprobar que funciona
- Seguir con el resto de las operaciones…

Vista: listarUsuarios



Recursos

- Spring Web MVC
- https://www.thymeleaf.org/documentation.html
- https://developer.mozilla.org/en-US/docs/Web/HTML
- Baeldung (ejemplos cortos, buscar Thymeleaf)
  - https://www.baeldung.com/full_archive
- https://getbootstrap.com/docs/
- https://fontawesome.com/docs/web/

Recursos Thymeleaf

- Thymeleaf por Daniel Fernández (creador)
  - https://youtu.be/GVq0uzpHYoQ?feature=shared
- Thymeleaf Fragments
  - https://www.thymeleaf.org/doc/articles/layouts.html
  - https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html#fragment-specification-syntax
  - https://www.baeldung.com/spring-thymeleaf-fragments