Desarrollo web con

Spring MVC

¿Qué es Spring MVC?

- Spring MVC utiliza una arquitectura de aplicaciones siguiendo el patrón de diseño MVC (**M**odel **V**iew **C**ontroller).
- Spring MVC es un framework web basado en Servlets que viene incluido en Spring Framework (spring-webmvc).
- Spring MVC esta diseñado siguiendo el patrón de diseño de **Front Controller**.
- En Spring MVC el Front Controller es mejor conocido como **DispatcherServlet**.

  Funciones

  - ✓ Enviar las peticiones (requests) a los manejadores (handlers) para que sean procesadas.
    - El default handler son los controladores (@Controller, @RequestMapping).
  - ✓ Encargado de resolver las vistas (views).
- Apartir de Spring Framework 3.0 se pueden crear RESTFul Web Services utilizando la anotación @**RestController** y @**PathVariable**.
- Basado en Spring IOC container (Inyección de Dependencias).
- Spring MVC se integra muy fácil con otros proyectos de Spring:
  - ✓ Nosotros integraremos **Spring Boot, Spring Data JPA, Spring Security, Spring REST, etc.**

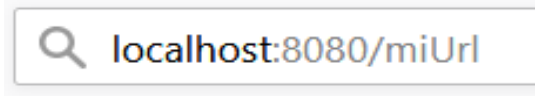
¿Qué es un Controlador en Spring MVC?

- Un Controlador (Controller) en Spring MVC es una clase normal a la cual se le agrega la anotación **@Controller** a nivel de la clase.
- En una aplicación web estos métodos principalmente están marcados con las anotaciones @GetMapping, @PostMapping, y @RequestMapping (Action Controller).
- Los métodos pueden tener cualquier nombre y deben regresar un String (nombre de la vista).
- Los métodos son ejecutados al ser invocados por medio de la URL especificada como parámetro en las anotaciones @GetMapping, @PostMapping, etc.

```
@Controller
public class HomeController {


 @GetMapping("/miUrl")
 public String mostrarHome() {
   // Mi lógica de negocio
   return "home";
 }

}
```



PETICIÓN TIPO GET

Cuando se utiliza el motor de plantillas **Thymeleaf** se buscará un archivo (vista) llamado **home.html** en el directorio: src/main/resources/templates

¿Qué es Thymeleaf? (1)

- Thymeleaf es un motor de plantillas para aplicaciones web desarrolladas con Java. Es algo similar a los JSPs, con algunas diferencias.
  - ✓ Página Oficial: www.thymeleaf.org
- Comúnmente utilizado con Spring Boot para generar vistas con código HTML para aplicaciones web.
- En un proyecto Spring boot, ya viene configurado Thymeleaf con valores por defecto al momento de agregar la dependencia.

Configuración

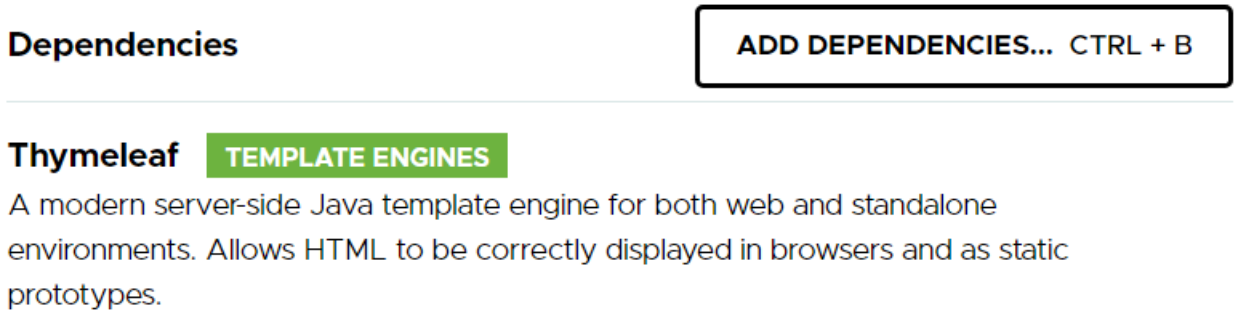
- Agregar la siguiente dependencia al archivo **pom.xml**:

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

¿Qué es Thymeleaf? (2)

Configuración

- Si creamos nuestro proyecto con **Spring Initializr:**



- Para utilizar **Thymeleaf** en un archivo HTML se debe agregar el namespace.

home.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Titulo</title>
</head>
<body>
  <h1 th:text="${mensaje}"></h1>
</body>
</html>
```

HomeController.java

```
@Controller
public class HomeController {

  @GetMapping("/")
  public String mostrarHome(Model model) {
    model.addAttribute("mensaje", "Hola Mundo");
    return "home";
  }

}
```

Iteraciones en Thymeleaf

- En Thymeleaf las iteraciones se pueden realizar con la expresión **th:each**
  - ✓ Similar a un for en Java.
- Esta expresión puede iterar sobre diferentes tipos de datos como:
  - ✓ List
  - ✓ Map
  - ✓ Iterable
  - ✓ Map

Ejemplo:

Vista(detalle.html)

```
<tr th:each="tmpEmp: ${empleos}">

    <td th:text="${tmpEmp}" />

</tr>
```

Controlador

```
@GetMapping("/detalle")
public String mostrarDetalle(Model model) {
        List<String> lista = new LinkedList<>();
        lista.add("Ingeniero de Sistemas");
        lista.add("Auxiliar de Contabilidad");
        model.addAttribute("empleos", lista);
        return "detalle";
}
```

Condicionales en Thymeleaf

➢Operador Elvis (?:)

El operador Elvis permite renderizar texto DENTRO de un elemento HTML, dependiendo de una expresión Booleana. Es muy similar al operador ternario en otros lenguajes de programación.

Ejemplo:

```
<td th:text="${usuario.estatus == 1} ? 'ACTIVO' : 'BLOQUEADO'" />
```

Valor renderizado, si la expresión es verdadera.

Valor renderizado, si la expresión es falsa.

➢if – unless

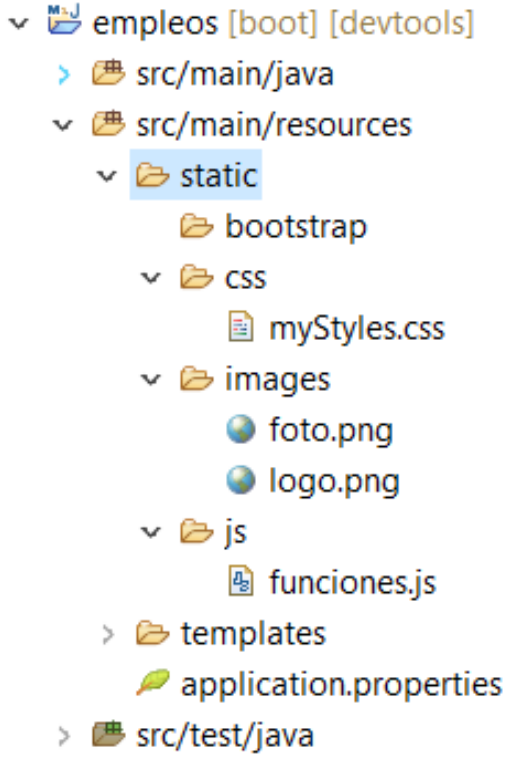
La expresión **if - unless** permite renderizar un elemento HTML, dependiendo de una expresión Booleana. Es muy similar a un **if-else** en otros lenguajes de programación.

Ejemplo:

```
<td>
   <span th:if="${alumno.genero == 'F'}"> Femenino </span>
   <span th:unless="${alumno.genero == 'F'}"> Masculino </span>
</td>
```

Urls relativas al ContextPath en Thymeleaf

- Las URLs relativas al ContextPath son las que son relativas al directorio raíz (ROOT) de una aplicacion web, una vez que están publicadas en el servidor.
- Las URLs relativas al ContextPath deben iniciar con "/" cuando vayamos a formar una URL para referenciar un recurso (imagenes, CSS, JS, PDF, etc) en nuestra aplicacion.
- En un proyecto web cuando se utiliza **Thymeleaf** como motor de plantillas, los recursos estáticos deben guadarse en el directorio **src/main/resources/static**



Ejemplos:

- Para incluir el archivo CSS myStyles.css en una vista se utilizaría la siguiente expresión:
  <link th:href="@{/css/myStyles.css}" rel="stylesheet">
- Para incluir el archivo Javascript funciones.js en una vista se utilizaría la siguiente expresión:
  <script th:src="@{/js/funciones.js}"></script>
- Para incluir la imagen foto.png en una vista, se utilizaría la siguiente expresión:
  <img th:src="@{/images/foto.png}" width="136" height="136" >

Para incluir archivos Javascript y CSS vía CDN (*Content Delivery Network*) se utiliza la sintáxis estándar (sin expresiones Thymeleaf).

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
```