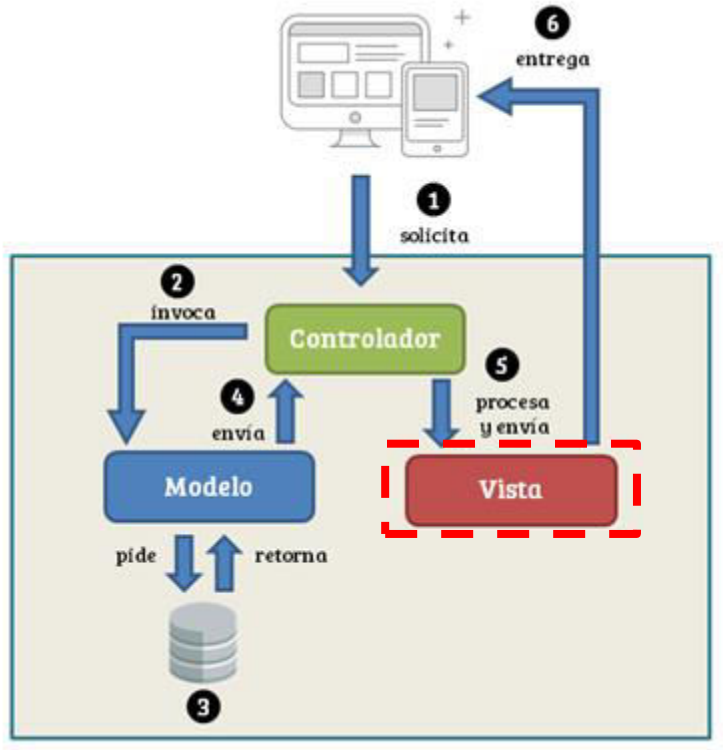
VISTAS
EN SPRING MVC

VISTA

Es la parte encargada de renderizar las plantillas que visualizará el usuario.



“

Spring MVC está diseñado para ser independiente de la tecnología de la vista. Podemos verlo como piezas de un puzzle que encajan. Así podemos seleccionar la tecnología más adecuada.

POSIBLES TECNOLOGÍAS EN LA VISTA

Thyemeleaf

Motor de plantillas que enfatiza el uso de plantillas naturales. Su integración con Spring es muy sencilla, e inmediata con Spring Boot.

FreeMarker

Apache FreeMarker es un motor de plantillas para generar contenido HTML. Integración sencilla con Spring.

Groovy Markup

Motor de plantillas para generar contenido XML-like. Requiere del uso de Groovy 2.3.1+

JSP + JSTL

Tecnología Java que permite crear páginas web dinámicas. A nivel de rendimiento es equivalente al uso de Servlets.

Tiles

Apache Tiles es un marco de desarrollo de plantillas que ha sido muy popular por su uso junto a Struts.

Otras (JSR-223)

Handlebars, Mustache, React, EJS, … o cualquier otro sistema que pueda correr sobre un motor de scripting JSR-233.

CARACTERÍSTICAS DE THYMELEAF

- Es un motor de plantillas: plantilla + modelo = resultado.

modelo

nombre = Pepe

apellidos=Pérez Pérez

plantilla

```
<div ...>
        <p>${nombre}</p>
        <p>${apellidos}</p>
</div>
```

resultado

```
<div ...>
        <p>Pepe</p>
        <p>Pérez Pérez</p>
</div>
```

CARACTERÍSTICAS DE THYMELEAF

- Natural Templating

```
<html>
<head>
  <title>Hola mundo</title>
</head>
<body>
        <p th:text="${saludo}">Hola Mundo</p>
</body>
</html>
```

Esta etiqueta es propia de HTML. El atributo no lo es, y el navegador lo descarta, pero no da error.

Si trabajamos con esta plantilla de forma estática (sin pasar por el motor), podemos ver un texto por defecto.

CARACTERÍSTICAS DE THYMELEAF

- *Procesador*: un objeto que aplica una transformación a un determinado artefacto (texto, etiqueta, comentario, …)
- **Dialectos: conjunto de procesadores.**
- Spring posee un dialecto propio (*SpringStandardDialect*).
- Nos permite utilizar SpEL (Spring Expression Language) en lugar del lenguaje de expresiones por defecto.

CÓMO CONFIGURAR THYMELEAF COMO MOTOR DE PLANTILLAS

- La forma más sencilla: Spring Boot + starter de Thymeleaf

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

¿QUÉ HACE SPRING BOOT POR NOSOTROS PARA CONFIGURAR THYMELEAF?

- Configura varios *beans*
  - *ViewResolver* (*ThymeleafViewResolver*) Encargado de resolver las vistas.
  - *TemplateEngine* (*SpringTemplateEngine*)

    Indica el dialecto Thymeleaf y habilita el lenguaje de expresiones EL.
  - *TemplateResolver* (*SpringResourceTemplateResolver*)

    Permite acceder físicamente a las plantillas, indicando ruta y sufijo

¿QUÉ HACE SPRING BOOT POR NOSOTROS PARA CONFIGURAR THYMELEAF?

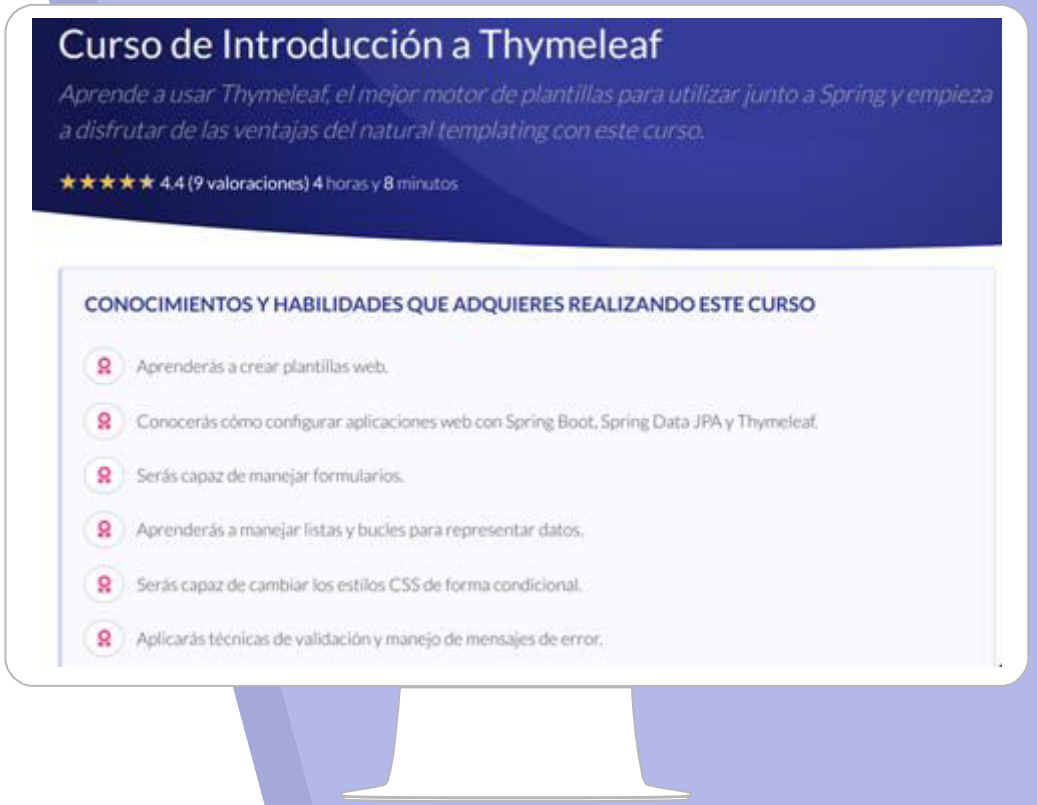
```java
@Configuration
public class ThymeleafConfiguration {

    @Bean
    public SpringResourceTemplateResolver templateResolver(){
        SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();
        templateResolver.setApplicationContext(this.applicationContext);
        templateResolver.setPrefix("/WEB-INF/templates/");
        templateResolver.setSuffix(".html");
        templateResolver.setTemplateMode(TemplateMode.HTML);
        templateResolver.setCacheable(true);
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine(){
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver());
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }

    @Bean
    public ThymeleafViewResolver viewResolver(){
        ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
        viewResolver.setTemplateEngine(templateEngine());
        viewResolver.setOrder(1);
        viewResolver.setViewNames(new String[] {".html", ".xhtml"});
        return viewResolver;
    }

}
```

CÓMO MODIFICAR LA CONFIGURACIÓN POR DEFECTO DE THYMELEAF

- *application.properties*
  - spring.thymeleaf.cache
  - spring.thymeleaf.check-template
  - spring.thymeleaf.check-template-location
  - spring.thymeleaf.enabled
  - spring.thymeleaf.encoding
  - spring.thymeleaf.mode
  - spring.thymeleaf.prefix
  - spring.thymeleaf.suffix

PROFUNDIZAR EN THYMELEAF

Accede a nuestro [Curso de Introducción a Thymeleaf] y profundiza en esta tecnología para sacar más partido a este curso.



ALGO PARA PRACTICAR

Un ejercicio para hacer por tu cuenta

CAMBIO DE PROPIEDADES DE THYMELEAF

- Prueba a *jugar* con el fichero *application.properties*, cambiando el valor de alguna de las propiedades de configuración.
- Por ejemplo, *spring.thymeleaf.cache=false* te permitirá realizar cambios en la plantilla y poder visualizarlos sin tener que *relanzar* el proyecto.

CAMBIO DE PROPIEDADES DE THYMELEAF

- Si la extensión *.html* no te gusta, puedes probar a cambiar el sufijo a *.htm*