Thymeleaf

Tipos de expresiones

Permite trabajar con varios tipos de expresiones:

- **Expresiones variables**: Son quizás las más utilizadas, como por ejemplo `${...}`
- **Expresiones de selección**: Son expresiones que nos permiten reducir la longitud de la expresión si prefijamos un objeto mediante una expresión variable, como por ejemplo `*{...}`
- **Expresiones de mensaje**: Que nos permiten, a partir de ficheros properties o ficheros de texto, cargar los mensajes e incluso realizar la internalización de nuestras aplicaciones, como por ejemplo `#{...}`
- **Expresiones de enlace**: Nos permiten crear URL que pueden tener parámetros o variables, como por ejemplo `@{...}`
- **Expresiones de fragmentos**: Nos van a permitir dividir nuestras plantillas en plantillas más pequeñas e ir cargándolas según las vayamos necesitando, como por ejemplo `~{...}`

Por defecto, cuando se trabaja con Thymeleaf se suele hacer con el **lenguaje de expresiones OGNL** (Object-Graph Navigation Language), aunque cuando trabajamos conjuntamente con Spring MVC podemos utilizar el SpEL (Spring Expression Language).

Expresiones variables

Algunos ejemplos de expresiones variables son:

- `${sesión.usuario.nombre}` => Podemos usar la notación de puntos para acceder a las propiedades de un objeto.
- `<span th:text="${libro.autor.nombre}">` => Uno de los atributos que podemos usar es th:text con diferentes etiquetas HTML, para poder mostrar, por ejemplo, el nombre del autor de un libro. También podemos navegar entre objetos.
- `<p th:text="${#numbers.formatCurrency(5)}">` => Algunas expresiones vienen ya predefinidas, como la función formatCurrency, que nos va a permitir formatear un numero como una moneda y le va a añadir la moneda del sistema por defecto.

- <td th:text="${myObject.myMethod()}"> => También podemos llamar a métodos definidos en nuestros propios objetos, lo vamos a poder hacer desde las plantillas.

Expresiones de selección

Las expresiones de selección nos permiten marcar un objeto y sobre el mismo evaluar algunas expresiones.
Por ejemplo, si queremos listar todas las propiedades de un libro, en lugar de usar expresiones variables, podemos prefijar el objeto book e ir después utilizando cada una de las propiedades del mismo de una manera más sencilla:

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

Expresiones de enlace

Sirven para construir URLs que podemos utilizar en cualquier tipo de contexto.
Podríamos utilizarlas para hacer enlaces para URLs que sean absolutas o relativas al propio contexto de la aplicación, al servidor, al documento, etc.
Estos son unos ejemplos:

```
<a th:href="@{/order/list}">...</a>
  <a href="{/myapp/order/list}">...</a>
<a th:href="@{order/details(id=${orderId})}">...</a>
  <a ahref="{myapp/order/details?id=23">...</a>
<a th:href="@{.../documents/report}">...</a>
<a th:href="@{~/contenidos/index}">...</a>
<a th:href="@{http://www.micom.es/index}">...</a>
```

Expresiones literales

Podemos utilizar expresiones literales, como son habituales en otros lenguajes de programación:

- **Textuales**: 'texto', 'otro texto',...
- **Numéricos**: 0, 23, 3.14,...

- **Booleanos**: true, false
- **Nulos**: null
- **Tokens**: uno, texto, main,...

Operadores

Podemos utilizar los operadores más habituales, como son:

- **Textuales**: + (concatenación)
- **Aritméticos**:
  - Binarios: +, -, *, /, %
  - Unitarios: -
- **Booleanos**:
  - Binarios: and, or
  - Unitarios: !, not
- **De comparación**: <, >, >=, <= (gt, gl, ge, le)
- **De igualdad**: ==, != (eq, ne)
- **Condicionales**:
  - (if) ? (then)
  - (if) ? (then) : else
  - (valor) ?: (valor por defecto)

Atributos básicos

Los atributos básicos más conocidos con los que nos podemos encontrar son:

- th:text: Permite reemplazar el texto de la etiqueta por el valor de la expresión que le demos.

```
<p th:text="${saludo}>saludo</p>
```

- th:each:Nos va a permitir repetir tantas veces como se indique o iterar sobre los elementos de una colección.

```
<li th:each="libro : ${libros}"
th:text="${libro.titulo}">El Quijote</li>
```

•th:class:Permite modificar las clases de un elemento de forma que podríamos modular el CSS de ese elemento HTML.

<p th:class=”${row.even}? ’even’: ’odd’”></p>