Desarrollo avanzado de aplicaciones II

Tema Nº3:JSP y Thymeleaf.

Indicador de logro Nº3:Construye interfaces gráficas utilizando HTML, Javascript y CSS, considerando el motor de plantillas de Thymeleaf.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3:**

JSP y Thymeleaf.

**Subtema 3.1:**

Diseño de formularios con páginas JSP.

**Ejemplos:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Crear la siguiente estructura para el proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el Controlador agregar el siguiente código:

Texto

Descripción generada automáticamente

Y en la página HTML agregar el siguiente código:

Texto

Descripción generada automáticamente

**Subtema 3.2:**

Introducción a Thymeleaf.

**Ejemplos:**

Thymeleaf es un motor de plantillas Java para aplicaciones, tanto web como standalone, construido sobre estándares HTML5, por lo que es compatible con la inmensa mayoría de navegadores. Entre sus principales características cabe destacar su flexibilidad y su alto rendimiento que, junto con su facilidad de aprendizaje y su poco intrusismo en el marcado lo convierten en una alternativa real a otras opciones más populares como JSP.

**Ventajas de Thymeleaf**

Permite realizar tareas que se conocen como natural templating. Es decir, como está basada en añadir atributos y etiquetas, sobre todo HTML, va a permitir que nuestras plantillas se puedan renderizar en local, y esa misma plantilla después utilizarla también para que sea procesada dentro del motor de plantillas. Por lo cual las tareas de diseño y programación se pueden llevar conjuntamente.

Es integrable con muchos de los frameworks más utilizados, como por ejemplo Spring MVC. Y está basado en el uso de nuevas etiquetas, de nuevos atributos.

**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).

**Subtema 3.3:**

JSP vs Thymeleaf.

**Ejemplos:**

**Ventajas de Thymeleaf**

Las ventajas que nos ofrece Thymeleaf son:

* Podemos prototipar sin necesidad de ejecutar la aplicación.
* Si nuestra aplicación está corriendo y solo modificamos las plantillas, no es necesario volver a desplegarla, cambiando la propiedad cache a false.
* El dialecto estándar, o el dialecto de Spring si trabajamos con Thymeleaf conjuntamente con Spring, es mucho más potente y rico en funcionalidades que la librería de etiquetas de JSTL.

**Ventajas de JSP**

Las ventajas que ofrece JSP son estas:

* Es una tecnología que tiene mucho más recorrido, porque lleva en el mercado muchos más años.
* El procesamiento de las plantillas o del contenido, es mucho más rápido que la velocidad de procesamiento que puede conseguir Thymeleaf, por lo que en un sitio donde la velocidad sea critica quizás sea planteable la posibilidad de usar JSP.

**Subtema 3.4:**

Directivas de flujos y bucles en HTML con Thymeleaf.

**Ejemplos:**

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:

Imagen que contiene Texto

Descripción generada automáticamente

**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.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**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.



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

Texto

Descripción generada automáticamente

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



**Subtema 3.5:**

Bootstrap con Fragments en Thymeleaf.

Los fragmentos de Thymeleaf son bloques de código que podemos definir para posteriormente poder reutilizarlos en las distintas páginas de nuestras aplicaciones. Los fragmentos se pueden definir en ficheros separados o agrupar varios dentro de un mismo fichero para tenerlos juntos como en una plantilla.

**Como se usa los fragmentos**

Para utilizar los fragments hay 2 opciones th:insert= y th:replace.

* Con th:insert="ruta/archivo::nombre\_fragmento" el código del fragmento se incluye dentro del elemento en el que se usa esta etiqueta, por ejemplo con <body th:insert="layouts/base::header"> se incluye el fragmet header dentro del body.
* Otra opción es usar th:replace="ruta/archivo::nombre\_fragmento" para reemplazar la etiqueta por el fragmento, si usamos <header th:replace="layouts/base::header"> </header> el resultado va a ser que esa etiqueta header se va a sustituir por el contenido del fragmento header en lugar de incluirlo dentro como en el caso anterior.

**Ejemplos:**

Añadimos BootStrap a nuestro proyecto.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Creamos un archivo layout.html y añadimos el siguiente componente de BootStrap

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Actividad:**

a) CUESTIONARIO TÉCNICO

* ¿Qué es Thymeleaf?
* Explique dos expresiones de variables de Thymeleaf.
* Explique las ventajas de Thymeleaf frente a JSP.
* Diseñe 2 formularios con Bootstrap y Thymeleaf.
* Cree dos controladores y añada funcionalidad a su criterio al proyecto.

b) CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_