Ejercicios de creación de librerías de tags

Índice

1 Gestor de tags para un tag que formatee números reales....2
2 Configuración y prueba de nuestro tag....2
3 (*) Iterador sobre una cadena....3

1. Gestor de tags para un tag que formatee números reales

En la plantilla tenemos una aplicación `jsp-sesion10`, y en su directorio de fuentes tenemos una clase Java llamada `es.ua.jtech.jsp.sesion10.libtags1.FormateaDoubleTag`. Dicha clase está vacía, pero deberá contener el código necesario para desarrollar un tag llamado `formateadouble`. Dicho tag admitirá dos atributos:

- Uno llamado `valor`, que contendrá un valor real
- Uno llamado `decimales`, que indicará un número de decimales concreto (un valor entero: 0, 1, 2... etc).

Lo que hará el tag será formatear el valor real que le pasamos en `valor`, para sacarlo en la página con tantos decimales como indiquemos en `decimales`. El tag no tendrá cuerpo.

Para hacer eso, podemos utilizar la clase `java.text.NumberFormat` de Java, que permite indicar con cuántos decimales queremos formatear un determinado número:

```
NumberFormat nf=NumberFormat.getInstance();
nf.setMaximumFractionDigits(2);
nf.setMinimumFractionDigits(2);
...
System.out.println (nf.format(2.34786));                  // Sacaría 2.35
System.out.println (nf.format(2.7));                          // Sacaría
2.70
```

Deberemos rellenar la clase, con los métodos `doStartTag()`, `doEndTag()` y los métodos para acceder y establecer el valor de los campos y atributos que se tengan. Podemos fijarnos en el tag `Saludo` visto en teoría como ejemplo.

2. Configuración y prueba de nuestro tag

Una vez hecha la clase del tag, vamos a probarla en una página JSP. Para ello:

- Creamos el fichero TLD (`libtags1.tld`) que dé soporte a la librería. Dicho fichero sólo tendrá información del tag que hemos creado, indicando su nombre, atributos, y características.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.  ...
<taglib>

        <tlibversion>1.0</tlibversion>
        <jspversion>1.1</jspversion>
        <shortname>lt1</shortname>
        <uri>libtags1</uri>
```

```
        <info>Ejercicio de libreria propia</info>

        <tag>
                <name>formateadouble</name>
                <tagclass>...</tagclass>
                <bodycontent>...</bodycontent>
                <info>Formatea un double con n decimales</info>
                <attribute>
                        <name>valor</name>
                        <required>...</required>
                        <rtexprvalue>...</rtexprvalue>
                </attribute>
                <attribute>
                        <name>decimales</name>
                        <required>...</required>
                        <rtexprvalue>...</rtexprvalue>
                </attribute>
        </tag>

</taglib>
```

Las etiquetas que tienen puntos suspensivos (`...`) deben rellenarse adecuadamente: nombre de la clase que implementa el tag (`tagclass`), tipo de contenido (`bodycontent`), y características de los atributos: el primero (`valor`) será requerido, y admitirá expresiones `<%=...%>`, y el segundo (`decimales`) no será requerido (tomará valor de 0 decimales, por defecto), y no admitirá dichas expresiones.

Copiamos dicho fichero en el directorio `WEB-INF` de la aplicación.

- Modificamos el fichero descriptor (`web.xml`) para añadir el fichero TLD anterior:

```
<taglib>
        <taglib-uri>libtags1</taglib-uri>
        <taglib-location>/WEB-INF/libtags1.tld</taglib-location>
</taglib>
```

- Creamos una página **pruebaformato.jsp**, que invoque al tag anterior:

```
<%@ taglib uri="libtags1" prefix="lt1"%>

...
        Numero con 3 decimales:
        <lt1:formateadouble valor="2.35879" decimales="3"/>
...
```

La llamamos con:

```
http://localhost:8080/appmistags/pruebaformato.jsp
```

Y debería devolver 2.359 para el caso anterior.

3. (*) Iterador sobre una cadena

Construid sobre la librería anterior una etiqueta llamada `iteraCadena` de tipo `BodyTag` (que permita iterar sobre su contenido), que tome como parámetro un atributo `cadena`, que sea un `String`. La etiqueta tendrá a su vez una subetiqueta llamada `carActual` que sacará por pantalla el carácter en la cadena correspondiente a la iteración actual. De esta forma, un código como:

```
<%@ taglib uri="libtags1" prefix="lt1"%>

...
        <lt1:iteraCadena cadena="miCad">
                <lt1:carActual/> -
        </lt1:iteraCadena>
...
```

debería sacar por pantalla: *m - i - C - a - d -*