Servlets y JSP

- Sesión 8: Introducción a las librerías de tags

Puntos a tratar

- Introducción a las librerías de tags
- Ejemplo de librería: request
- Ejemplo de librería: dbtags
- Otras consideraciones

Definición

- Una librería de tags (*taglib*) es un conjunto de etiquetas HTML personalizadas que permiten encapsular acciones mediante código Java
- Cada etiqueta define un nexo entre la página JSP donde se coloca y el código Java subyacente que se ejecutará cuando se procese dicha etiqueta

```
<%@taglib uri="ejemplo" prefix="ej" %>
<html>
<body>
<h1>Ejemplo de librerias de tags</h1>
<ej:mitag>Hola a todos</ej:mitag>
</body>
</html>
```

Gestor de tags

- Cada tag tiene asociada una clase (gestor) con el código que se ejecuta al utilizar el tag
- La clase implementa una de las dos interfaces de *javax.servlet.jsp.tagext*:
  - *Tag*: para tags simples, que no manipulan su cuerpo
  - *BodyTag*: para tags que manipulan su contenido (cuerpo)
- La clase define una serie de métodos que son los que se van llamando a medida que se procesa el tag.

Gestor de tags (II)

- *doStartTag( )* se llama al iniciarse el tag, y devuelve:
  - EVAL_BODY_INCLUDE: para procesar el cuerpo de un tag de tipo *Tag*
  - EVAL_BODY_TAG: para procesar el cuerpo de un tag de tipo *BodyTag*
  - SKIP_BODY: para no procesar el cuerpo del tag

- *doEndTag( )* se llama cuando se termina el tag, y devuelve:
  - EVAL_PAGE: para continuar evaluando la página
  - SKIP_PAGE: para dejar de evaluar la página

Gestor de tags (III)

- Para los gestores de tipo *BodyTag* existen los métodos *doInitBody( )* y *doAfterBody( )* para poder manipular el cuerpo del tag, al procesarlo (y poder hacer iteraciones, o cosas similares)

- Una vez se tiene definido el gestor, se coloca como clase Java en el directorio *WEB-INF/classes* de la aplicación, respetando paquetes y subpaquetes
- Otra alternativa es empaquetar todos los gestores en una librería (fichero JAR), y ponerlo en *WEB-INF/lib*

Descriptor de la librería de tags

- Es un fichero XML con extensión .TLD en general que contiene directivas de descripción de una librería de tags
- Para utilizar un descriptor de librería, se coloca en el fichero descriptor (*web.xml*) de la aplicación:

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

Contenido del descriptor

- El fichero TLD tendrá un contenido como:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems..." ...>
<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>pt</shortname>
    <uri>pruebatags</uri>
    <info>Libreria de prueba</info>
    <tag>
          <name>prueba</name>
          <tagclass>Prueba</tagclass>
          <bodycontent>empty</bodycontent>
          <info>Tag de prueba</info>
          <attribute>
                    <name>nombre</name>
                    <required>false</required>
                    <rtexprvalue>false</rtexprvalue>
          </attribute>
    </tag>
</taglib>
```

Contenido del descriptor(II)

- *tlibversion* indica la versión de la librería
- *jspversion* indica a partir de qué versión de JSP es compatible
- *shortname* es un nombre corto para referenciarla
- *uri* es un identificador para la librería
- *info* es una cadena descriptiva de la librería

Contenido del descriptor(III)

- Para cada tag que haya, tendremos una etiqueta *tag* con:
  - *name*: nombre del tag
  - *tagclass*: nombre de la clase que implementa el tag
  - *bodycontent*: tipo de cuerpo del tag: *empty* (sin cuerpo), *tagdependent* (cuerpo evaluado por la propia clase) o *jsp* (cuerpo evaluado por el contenedor JSP)
  - *info*: información sobre el tag
  - *attribute*: para cada atributo que necesite el tag:

    name: nombre del atributo

    required: si es requerido (*true*) o no *(false*)

    rtexprvalue: a *true* indica que podemos colocar expresiones <%=...%> (por defecto es *false*)

Uso de taglibs en páginas JSP

- Primero copiamos el fichero TLD y las clases donde toque en *WEB-INF*
  - Muchas librerías vienen en un fichero JAR que ya incorpora el TLD
- Después indicamos en *web.xml* dónde está el TLD
  - No necesario en últimas versiones de JSP/servlets

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

- Finalmente incluimos la librería al principio de la página JSP, y la usamos con el prefijo indicado:

```
<%@taglib uri="prueba" prefix="pr" %>
<html>
...
<pr:hola/>
...
```

Introducción

- *request* es una librería desarrollada por el proyecto Jakarta para acceder al contenido de una petición HTTP

- Encontraremos información y últimas versiones en:

http://jakarta.apache.org/taglibs/doc/request-doc/intro.html

Uso de la librería

- Tiene un fichero TLD (*request.tld*) que hay que copiar en nuestro directorio *WEB-INF*
- También hay un fichero *request.jar* con los tags implementados, que se copia en *WEB-INF/lib*
- Finalmente, al principio de las páginas JSP donde la usemos, pondremos:

```
<%@taglib uri="http://jakarta.apache.org/taglibs/request-1.0"
   prefix="req" %>
```

Información general de la petición

- El tag *request* Obtiene información sobre la petición HTTP

```
<req:request id="req">

   Tamaño peticion:
   <jsp:getProperty name="req" property="contentLength"/>

   Dirección cliente:
   <jsp:getProperty name="req" property="remoteHost"/>

</req:request>
```

Toma de parámetros

- El tag *parameter* obtiene el valor de un parámetro dado

```
<req:parameter name="param1"/>
```

- El tag *parameters* recorre todos los parámetros

```
<req:parameters id="id1" >
   Nombre:<jsp:getProperty name="id1" property="name"/>
   Valor: <jsp:getProperty name="id1" property="value"/>
</req:parameters>

<req:parameters id="id1" name="param1">
   Valor: <jsp:getProperty name="id1" property="value"/>
</req:parameters>
```

Toma de parámetros (II)

- El tag *parameterValues* es interno a *parameters* y obtiene los valores del parámetro actualmente explorado

```
<req:parameters id="id1" name="param1">
   <req:parameterValues id="id2">
        Valor:<jsp:getProperty name="id2" property="value"/>
   </req:parameterValues>
</req:parameters>
```

- El tag *equalsParameter* compara el valor de un parámetro

```
<req:equalsParameter name="param1" match="hola" value="false">
   El parámetro no coincide con 'hola'
</req:equalsParameter>
```

- El tag *existsParameter* comprueba si existe un parámetro

```
<req:existsParameter name="param1" value="false">
   El parámetro 'param1' no existe
</req:existsParameter>
```

Cookies

- Los tags *cookie*, *cookies*, *equalsCookie* y *existsCookie* son similares a *parameter, parameters, equalsParameter, existsParameter*, pero para recorrer y explorar cookies

```
<req:cookies id="id1" >
   Nombre:<jsp:getProperty name="id1" property="name"/>
   Valor:<jsp:getProperty name="id1" property="value"/>
   Max.edad:<jsp:getProperty name="id1" property="maxAge"/>
</req:cookies>
```

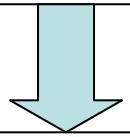
Cabeceras

- Los tags *header*, *headers, headerValues*, *equalsHeader* y *existsHeader* son similares a *parameter, parameters, parameterValues, equalsParameter* y *existsParameter*, pero para recorrer y explorar cabeceras HTTP de la petición

```
<req:header name="nombre"/>
```

Ejemplo de uso

```
<html><body>
   <form action="request.jsp">
        <input type="text" name="nombre">
        <input type="text" name="descripcion">
        <input type="submit" value="Enviar">
   </form>
</body></html>
```



```
<%@taglib uri="http..." prefix="req" %>
<html><body>
   Nombre:<req:parameter name="nombre"/>
   <req:existsParameter name="descripcion" value="true">
        Descripcion:<req:parameter name="descripcion"/>
   </req:existsParameter>
</body></html>
```

Introducción

- *dbtags* es otra librería desarrollada por el proyecto Jakarta para manipular bases de datos

- Encontraremos información y últimas versiones en:

http://jakarta.apache.org/taglibs/doc/dbtags-doc/index.html

Uso de la librería

- Tiene un fichero TLD (*dbtags.tld*) que hay que copiar en nuestro directorio *WEB-INF*
- También hay un fichero *dbtags.jar* con los tags implementados, que se copia en *WEB-INF/lib*
- Finalmente, al principio de las páginas JSP donde la usemos, pondremos:

```
<%@taglib uri="http:// jakarta.apache.org/taglibs/dbtags"  prefix="sql" %>
```

Conexión / Desconexión

- El tag *connection* conecta con una base de datos con los parámetros indicados

```
<sql:connection id="con1">
   <sql:url>jdbc:mysql://localhost/prueba</sql:url>
   <sql:driver>org.gjt.mm.mysql.Driver</sql:driver>
   <sql:userId>root</sql:userId>
   <sql:password>mysql</sql:password>
</sql:connection>
```

- El tag *closeConnection* cierra una conexión con un *id* determinado

```
<sql:closeConnection conn="con1">
```

Sentencias SQL

- El tag *statement* permite construir una sentencia que lanzar contra la base de datos (una vez conectados). Dentro, podemos usar los subtags:
  - *query*: para indicar la sentencia a ejecutar
  - *execute*: para ejecutar la sentencia, si es INSERT, UPDATE o DELETE
  - *resultSet*: para ejecutar y recorrer los resultados obtenidos, si es SELECT. El contenido de este tag se ejecuta una vez para cada registro obtenido.
    - Con *getColumn* obtendremos el valor de una columna concreta del registro
    - Con *wasEmpty* diremos qué hacer si no hay registros
    - Con *wasNotEmpty* diremos qué hacer si hay registros

Sentencias SQL (II)

```
<sql:statement id="s1" conn="con1">
   <sql:query>
   INSERT INTO datos(nombre, desc) VALUES ('n1', 'desc1')
   </sql:query>
   <sql:execute/>
</sql:statement>

<sql:statement id="s2" conn="con1">
   <sql:query>SELECT * FROM datos</sql:query>
   <sql:resultSet id="rs1">
        Nombre: <sql:getColumn colName="nombre"/>
        Descripcion: <sql:getColumn position="2"/>
        <br>
   </sql:resultSet>
   <sql:wasEmpty>No hay resultados</sql:wasEmpty>
</sql:statement>
```

Uso de los tags

- Se pueden colocar tags en diferentes lugares de nuestra página: por ejemplo, para obtener un parámetro de una *query* o una URL para un enlace:

```
<sql:statement id="s1" conn="con1">
   <sql:query>
        SELECT * FROM datos WHERE nombre =
        '<req:parameter name="nombre"/>'
   </sql:query>
   ...
</sql:statement>
```

```
<a href="http://<req.parameter name="url"/>">Enlace</a>
```

Comunicación entre JSP y taglibs

- Si queremos utilizar código JSP en una taglib, podemos hacerlo (si admite código JSP en sus atributos)

```
<sql:resultSet id="rs1" >
   <% for (int i = 0; i <= 3; i++) { %>
   Valor: <sql:getColumn position="<%=i%>"/>
   <% } %>
</sql:resultSet>
```

Comunicación entre JSP y taglibs (II)

- Si queremos utilizar una taglib dentro de código JSP es más complejo. Algunos tags disponen de métodos accesibles, y en otras hay que acceder a través del contexto

```
<sql:connection id="con1">
   ...
</sql:connection>
<% if (con1.getClosed()) { ... %>


<sql:getColumn colName="nombre" to="nombrePer"/>
...
<% if (pageContext.getAttribute("nombrePer").equals("pepe"))
   { ... }
%>
```

Más información

- Para consultar otras posibles librerías de tags, y últimas actualizaciones y novedades del proyecto Jakarta:

http://jakarta.apache.org/taglibs/index.html