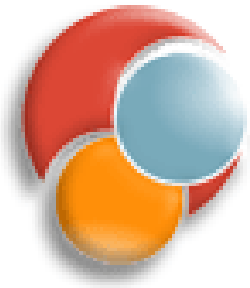

Componentes web

Sesión 8: Creación de *tags* propias

Opciones para crear tags propias

- Desde las primeras versiones de JSP, los usuarios pueden crear sus tags propias
  - Desarrollando una clase Java que implemente las operaciones necesarias, pero el API a implementar es complejo
- A partir de JSP 2.0, se dan dos nuevas opciones:
  - **Tag files**: tags como bloques de JSP, es decir, similares a un include
  - **Simple tags**: clases java, pero con un API más simple que el tradicional

Tag files

- Archivos que encapsulan bloques de JSP reutilizables
  - Por convenio, se les da extensión .tag
  - El nombre de la etiqueta será el del archivo

*(archivo WEB-INF/tags/**copyright.tag**)*

```
Copyright JTECH 2008
```

(página JSP)

```
<%@ taglib prefix="jtech" tagdir="/WEB-INF/tags" %>
<jtech:copyright/>
```

Paso de parámetros

- Los parámetros se llaman aquí *atributos* y se pasan con la directiva attribute
  - required: obligatorio o no
  - rtexprvalue: *true* indica que se puede usar EL

(archivo WEB-INF/tags/copyright.tag)

```
<%@ attribute name="anyo" required="true" rtexprvalue="false"%>
Copyright JTECH ${anyo}
```

(página JSP)

```
<%@ taglib prefix="jtech" tagdir="/WEB-INF/tags" %>
<jtech:copyright anyo="2008"/>
```

Parámetros de tipo variable

- Interesante del ejemplo
  - type: tipo de datos del parámetro
  - Se pueden usar scriptlets en el .tag
  - En .tag no se usa <%@import %> sino <%@tag import %>

(archivo WEB-INF/tags/copyright.tag)

```
<%@ tag import="java.util.Date" import="java.text.SimpleDateFormat"%>
<%@ attribute name="fecha" required="true" type="java.util.Date" %>
<% SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
   out.print(sdf.format(fecha)); %>
```

(página JSP)

```
<%@ taglib prefix="jtech" tagdir="/WEB-INF/tags"%>
<jtech:fmtFecha fecha="<%= new java.util.Date() %>"/>
```

Etiquetas con contenido

- La acción **<jsp:body/>** obtiene el contenido del cuerpo del tag
  - El atributo var de la acción nos permite guardarlo en request, session o application.

(archivo WEB-INF/tags/copyright.tag)

```
<jsp:doBody var="texto" scope="request"/>
<% String texto = (String) request.getAttribute("texto");
   out.print(texto.toUpperCase()); %>
```

(página JSP)

```
<%@ taglib prefix="test" tagdir="/WEB-INF/tags"%>
<test:capitalizar>Esto es un texto que la etiqueta debería pasar a
   mayúsculas</test:capitalizar>
```

Simple tags

- Las *tags* se implementan como clases Java
  - Código “más limpio”
  - La clase debe implementar las operaciones del interface SimpleTag
    - La versión “clásica” necesitaba un interface mucho más complejo
- Pasos
  - Implementar la clase Java con el *tag*
  - Crear un descriptor de despliegue (.tld)

Paso 1: Implementar la clase Java

- La clase debe implementar el interface SimpleTag

```
package javax.servlet.jsp.tagext;
public interface SimpleTag extends JspTag {
    public void doTag() throws JspException,
                                    IOException;
    public JspTag getParent();
    public void setJspBody(JspFragment jspBody);
    public void setJspContext(JspContext jspContext);
    public void setParent(JspTag parent);
}
```

El “ciclo de vida” del *tag*

- Cuando el servidor encuentra nuestro *tag*, va llamando a sus métodos en cierto orden
  1. **setJspContext**: nos pasa el *JspContext,* que nos da acceso a *request, session, application* y al *JspWriter* con la salida
  2. **setParent**: quién es el tag "padre".

     Esto permite anidar tags propios de manera que "colaboren" entre sí, modificando su comportamiento y pasándose información cuando están anidados.
  3. **setJspBody**: nuestro tag recibe también el cuerpo que contiene, en forma de objeto de la clase *JspFragment*
  4. **doTag**: es el método más importante. Indica que "es hora de hacer el trabajo".

“Ejemplo simple” de simple tag

- *SimpleTagSupport* aporta implementaciones “vacías” de los métodos de *SimpleTag*, permitiéndonos acortar el código
- Nótese el uso que hacemos del JSPContext

```
package jtech;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class TestSimpleTag extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
     SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
     getJspContext().getOut().write(sdf.format(new Date()));
  }
}
```

Paso 2: Descriptor de despliegue

- Archivo WEB-INF/tlds/test.tld

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-
   jsptaglibrary_2_1.xsd" version="2.1">
 <description> Ejemplo de simple tag </description>
 <jsp-version>2.1</jsp-version>
 <tlib-version>1.0</tlib-version>
 <short-name>test</short-name>
 <uri>http://www.jtech.ua.es/ayto/test</uri>
 <tag>
   <name>test</name>
   <tag-class>jtech.TestSimpleTag</tag-class>
   <body-content>empty</body-content>
   <description> Es una prueba tonta, en realidad </description>
 </tag>
</taglib>
```

Usar el *tag*

Página JSP

```
<%@ taglib prefix="jtech" uri="WEB-INF/tlds/test.tld"%>
<jtech:testTag/>
```

Archivo WEB-INF/tlds/test.tld

```
<taglib …
 <description> Ejemplo de simple tag </description>
 …
 <short-name>test</short-name>
 <uri>http://www.jtech.ua.es/ayto/test</uri>
 <tag>
   <name>testTag</name>
   <tag-class>jtech.TestSimpleTag</tag-class>
   <body-content>empty</body-content>
   <description> Es una prueba tonta, en realidad </description>
 </tag>
</taglib>
```

Atributos en el *tag*

- Por cada atributo necesitamos getter/setter

```
public class TestSimpleTag extends SimpleTagSupport {
   private Date fecha;

   public void doTag() throws JspException, IOException {
     SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
     getJspContext().getOut().write(sdf.format(fecha));
   }
   public Date getFecha() {
     return fecha;
   }
   public void setFecha(Date fecha) {
     this.fecha = fecha;
   }
}
```

Atributos en el tag (II)

- En el .tld definimos los atributos

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-
   jsptaglibrary_2_1.xsd" version="2.1">
 <description> Ejemplo de simple tag </description>
 <jsp-version>2.1</jsp-version>
 <tlib-version>1.0</tlib-version>
 <short-name>test</short-name>
 <uri>http://www.jtech.ua.es/ayto/test</uri>
 <tag>
   <name>test</name>
   <tag-class>jtech.TestSimpleTag</tag-class>
   <body-content>empty</body-content>
   <description> Es una prueba tonta, en realidad </description>
   <attribute> <name>fecha</name> <required>true</required>
   <rtexprvalue>true</rtexprvalue> </attribute>
 </tag>
</taglib>
```