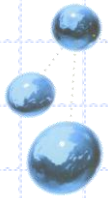


JAVA CUSTOM TAGS

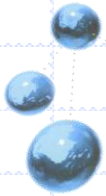
Creando tags personalizados

Profesores:

- Andrés Farías

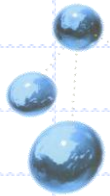
¿Qué es JSTL y cómo usarlo?

Introducción



JSTL

- Objetivo:
  - ✓ Simplificar y agilizar el desarrollo de aplicaciones web
- 3ra iteración después de servlets y JSPs
- Sirven para la generación dinámica de páginas web

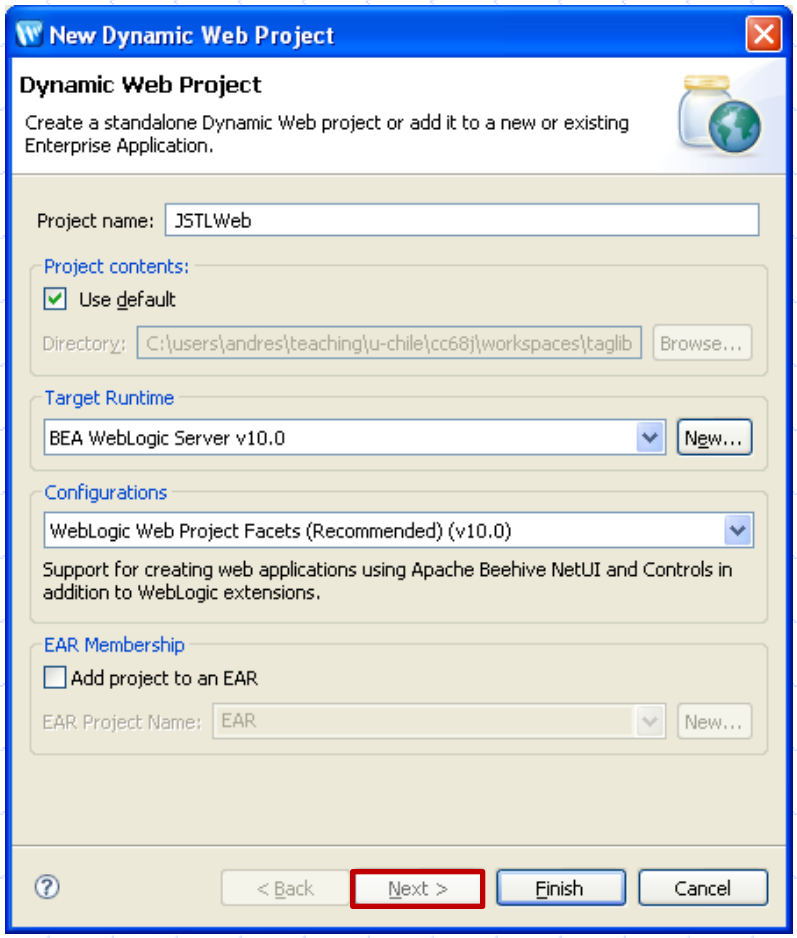


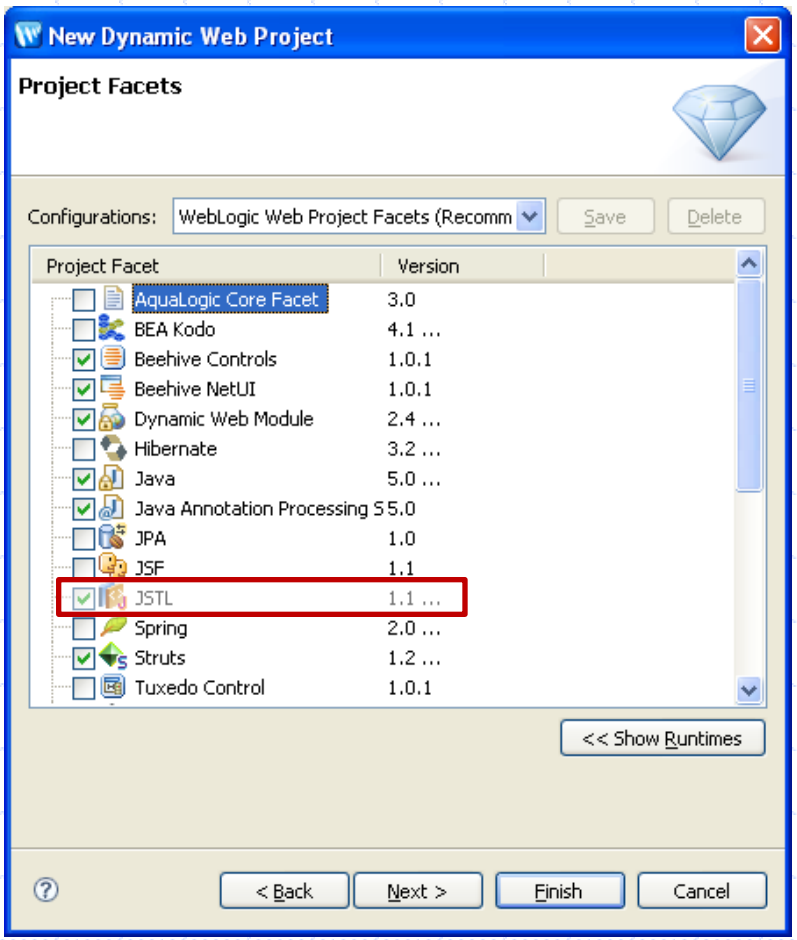
Utilizar JSTL

- Bajar JSTL 1.1 de:
  - ✓ http://cvs.apache.org/builds/jakarta-taglibs/nightly/jakarta-taglibs-20050216.zip
  - ✓ Copiar archivos **`standard.jar`** y **`jstl.jar`** a **`%TOMCAT_HOME%\common\lib`**
- http://jakarta.apache.org/taglibs/
- JSTL en WebLogic.
  - ✓ JSTL viene incorporado en las librerías.
  - ✓ Utilizando el wizard de proyectos dinámicos se puede seleccionar como librería.

Utilizando JSTL

En un proyecto con BEA Workshop





Utilizando JSTL

En un proyecto con BEA Workshop

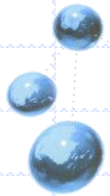
weblogic.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<wls:weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
  <wls:context-root>JSTLWeb</wls:context-root>
  <wls:library-ref>
    <wls:library-name>beehive-controls-1.0.1-10.0-war</wls:library-name>
    <wls:specification-version>1.0</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
    <wls:library-name>jstl</wls:library-name>
    <wls:specification-version>1.1</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
    <wls:library-name>struts-1.2</wls:library-name>
    <wls:specification-version>1.2</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
    <wls:library-name>weblogic-controls-10.0-war</wls:library-name>
    <wls:specification-version>10.0</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
    <wls:library-name>wls-commonslogging-bridge-war</wls:library-name>
    <wls:specification-version>1.0</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
    <wls:library-name>beehive-netui-1.0.1-10.0</wls:library-name>
    <wls:specification-version>1.0</wls:specification-version>
    <wls:exact-match>true</wls:exact-match>
  </wls:library-ref>
  <wls:library-ref>
```

Design | Source

JSTL 1.0 vs JSTL 1.1

- JSTL 1.1 es una pequeña mejora de JSTL 1.0 creada para alinear JSTL con JSP 2.0.
- Antes había una versión de cada librería dependiendo de si utilizabamos expresiones EL o Java, ahora es la misma librería.
  - ✓ Ha cambiado el nombre de los identificadores de las librerías, se ha añadido un elemento del path **/jsp** a todos ellos



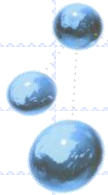
Usar JSTL en una aplicación Web

- Para cualquier aplicación web desde la cual quieres usar JSTL, copiar los ficheros **.tld** al directorio **WEB-INF** de tu aplicación web.
- Edita el **web.xml** de tu aplicación web añadiendo las siguientes entradas

```
<taglib>
  <taglib-uri>http://java.sun.com/jsp/jstl/fmt</taglib-uri>
  <taglib-location>/WEB-INF/fmt.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
  <taglib-location>/WEB-INF/c.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>http://java.sun.com/jsp/jstl/sql</taglib-uri>
  <taglib-location>/WEB-INF/sql.tld</taglib-location>
</taglib>

<taglib>
  <taglib-uri>http://java.sun.com/jsp/jstl/x</taglib-uri>
  <taglib-location>/WEB-INF/x.tld</taglib-location>
</taglib>
```

Características

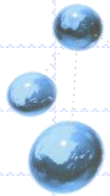
- Las páginas JSTL son también páginas JSP. JSTL es un superconjunto de JSP.
- JSTL provee un conjunto de cinco librerías estándar:
  - ✓ Core
  - ✓ Internationalization/format
  - ✓ XML
  - ✓ SQL y
  - ✓ Funciones
- Además JSTL define un nuevo lenguaje de expresiones llamado EL, que ha sido luego adoptado por JSP 2.0
- Una etiqueta JSTL corresponde a una acción; llamándolas acción nos indica que añaden comportamiento dinámico a una, de otra manera, página estática.



JSTL y EL

Soporte para EL (1/2)

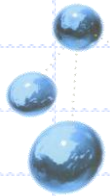
- El lenguaje de expresiones EL simplemente define un poderoso mecanismo para expresar expresiones simples en una sintáxis muy sencilla.
  - ✓ Es algo entre JavaScript y Perl.
  - ✓ Su combinación con las etiquetas de las 4 librerías antes mencionadas proveen mucha flexibilidad y poder para el desarrollo de páginas dinámicas.
- En EL las expresiones están delimitadas por **${ }**.



JSTL y EL

Soporte para EL (2/2)

- Algunos ejemplos del uso de EL son:
  - ✓ `${anExpression}`
  - ✓ `${aList[4]}`
  - ✓ `${aList[someVariable]}` → *acceso a un elemento de una colección*
  - ✓ `${anObject.aProperty}` → *acceso a la propiedad de un objeto*
  - ✓ `${anObject["aPropertyName"]}` → *entrada en un mapa con propiedad* `aPropertyName`
  - ✓ `${anObject[aVariableContainingPropertyName]}`
- Existen una serie de variables implícitas definidas en EL:
  - ✓ `pageContext`: *el contexto del JSP actual*
  - ✓ `pageScope`, `requestScope`, `sessionScope`, y `applicationScope`: *colecciones de mapas que mapean nombres de variables en esos contextos a valores*
  - ✓ `param` y `paramValues`: *parámetros pasados con la petición de la página, lo mismo que en JSP*
  - ✓ `header` y `headerValues`: *cabeceras pasadas en la petición de la página*
  - ✓ `cookie`: *mapa que mapea nombres de cookies a los valores de las mismas*



JSTL Tag Libraries

Resumen

Librería	URI	Prefix
Core	`http://java.sun.com/jsp/jstl/core`	`c`
Internationalization I18N	`http://java.sun.com/jsp/jstl/fmt`	`fmt`
SQL/DB support	`http://java.sun.com/jsp/jstl/sql`	`sql`
Procesamiento XML	`http://java.sun.com/jsp/jstl/xml`	`x`
Functions	`http://java.sun.com/jsp/jstl/functions`	`fn`

Uso de las librerías JSTL en un JSP

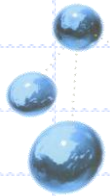
Importe manual

- La siguiente directiva ha de incluirse al comienzo de la página:

```
<%@ taglib prefix="c" uri=http://java.sun.com/jsp/jstl/core %>
```

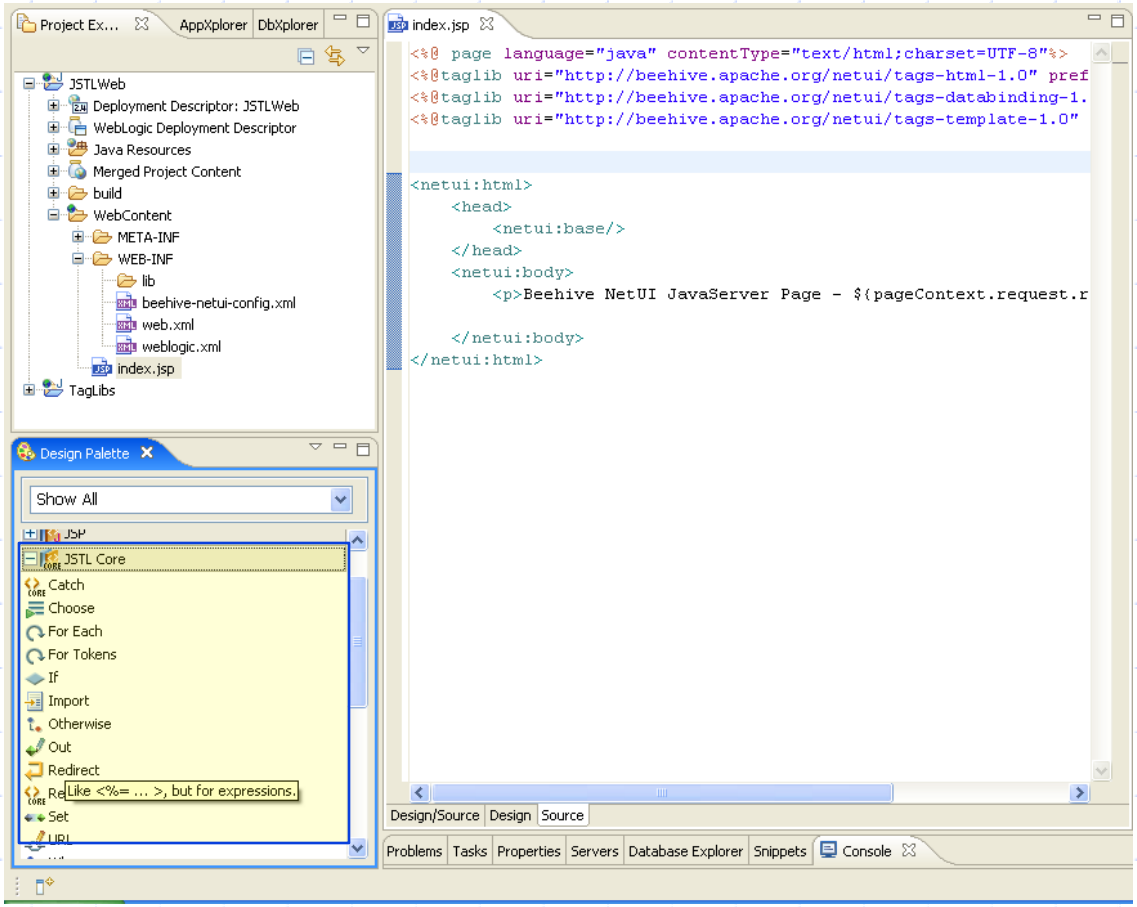
- Para utilizar una etiqueta de una librería simplemente se ha de preceder con el prefijo de la librería utilizada:

```
<c:out value="${anExpression}"/>
```



Uso de las librerías JSTL en un JSP

Importe con Design Palette



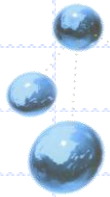
- El Design Palette provee acceso a las librerías JSTL.
- Dada una página JSP, basta con arrastrar algún TAG de la librería para que Workshop lance un asistente e importe la librería y la acción a la página.

Librerías de JSTL

Librería Core

La librería de etiquetas Core

- Permiten llevar a cabo las siguientes acciones:
  - ✓ Visualizar/asignar valores y manejar excepciones
  - ✓ Control de flujo
    - Condicion simple (if)
    - Selección condicional múltiple (choose)
  - ✓ Otras acciones de utilidad



TagLib Core
Visualizar/asignar valores

- Para visualizar valores utilizamos:

```
<c:out: value="${applicationScope.product.inventoryCount}"
   escapeXml="true" default="0"  /> of those items in stock.
```

  - ✓ **escapeXml** indica si hay que aplicar códigos de escape a los caracteres <, >, & y .
- Asignar una variable en una página:

```
<c:set var="customerID" value="$param.customerNumber"
   scope="session" />
```

  - ✓ **scope** indica el contexto en el que se define la variable
  - ✓ También podemos asignar el contenido de una etiqueta a una variable:

```
<c:set var="cellContents">
   <td>
     <c:out value="${myCell}"/>
   </td>
</c:set>
```

TagLib Core

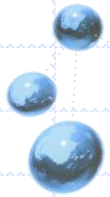
Manejar Excepciones

- Normalmente en un JSP o incluimos un bloque **try/catch** o usamos la directiva **errorPage**:

```
<c:catch>
Los tags aquí pueden lanzar excepciones
</c:catch>
```

- Para borrar una variable se puede utilizar

```
<c:remove var="myVar">
```



Control de flujo con Core

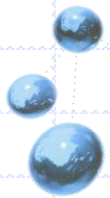
If y choose

- Para llevar a cabo simples condiciones (**c:if**):

```
<c:if test="${status.totalVisits == 1000000}" var="visits">
   You are the millionth visitor to our site!  Congratulations!
</c:if>
```

- El switch de un lenguaje de programación se puede emular con **c:choose**:

```
<c:choose>
<c:when test="${item.type == 'book'}">
...
</c:when>
<c:when test="${item.type == 'electronics'}">
...
</c:when>
<c:otherwise>
...
</c:otherwise>
</c:choose>
```

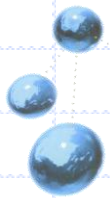


Control de flujo con Core

Foreach

- Para iterar sobre una colección se define **c:foreach**. Se pueden especificar índice de comienzo, final e incremento con los atributos **begin**, **end** y **step**.

```
<table>
   <c:forEach var="name" items="${customerNames}">
        <tr><td><c:out value="${name}"/></td></tr>
   </c:forEach>
</table>
```

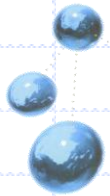


Control de flujo con Core

Fortokens

- Funcionalidad similar a **StringTokenizer** puede ser obtenida en JSTL con **c:forTokens**:

```
<table>
   <c:forTokens items="47,52,53,55,46,22,16,2"
   delims="," var="dailyPrice">
   <tr><td><c:out value="${dailyPrice}"/></td></tr>
   </c:forTokens>
</table>
```



Ejemplo con JSTL Core

Listar todos los parámetros pasados a una petición

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
<head>
<title>Parameter Listing Example</title>
</head>

<b>Parameter values passed to this page for each parameter: </b>
<table border="2">
<c:forEach var="current" items="${param}">
    <tr>
    <td><b><c:out value="${current.key}" /></b></td>
    <c:forEach var="aVal" items="${paramValues[current.key]}">
     <td>
       <c:out value="${aVal}" />
     </td>
    </c:forEach>
    </tr>
</c:forEach>
</table>
</body>
</html>
```

Otras acciones con Core

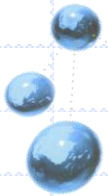
Url, import, redirect

- Para codificar URLs se puede utilizar **c:url**:

```
<c:url value="http://acme.com/exec/register" var="myUrl">
    <c:param name="name" value="${param.name}"/>
    <c:param name="country" value="${param.country}"/>
</c:url>
<a href='<c:out value="${myUrl}"/>'>Register</a>
```

- Se pueden importar otros JSPs o incluso otros recursos en una URL arbitraria usando **c:import** (análogo to **jsp:include**).

- Para manejar redireccionamiento se puede utilizar la etiqueta **c:redirect**



Nuevo soporte para el

EL extendido

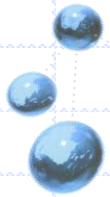
La nueva librería de funciones

- Cuando **EL** se migró de la especificación **JSTL** a **JSP**, se le añadió una nueva funcionalidad: la invocación a funciones.

- Su uso es trivial: **nombre-func(lista-params)**

```
<%@ taglib prefix="fn"
    uri="http://java.sun.com/jsp/jstl/functions" %>
${fn:length(myCollection)}
```

- Contiene un grupo de funciones utilizadas comúnmente en la programación de páginas.



EL extendido

Las funciones (1/2)

Función	Descripción
`fn:contains(string, substring)`	Devuelve true si el string contiene substring.
`fn:containsIgnoreCase(string, substring)`	Devuelve true si el string contiene a substring, ignorando capitalización.
`fn:endsWith(string, suffix)`	Devuelve true si el string termina con suffix.
`fn:escapeXml(string)`	Devuelve el string con todos aquellos caracteres con especial significado en XML y HTML convertidos a sus códigos de escape pertinentes.
`fn:indexOf(string, substring)`	Devuelve la primera ocurrencia de substring en el string.
`fn:join(array, separator)`	Devuelve un string compuesto por los elementos del array, separados por separator.
`fn:length(item)`	Devuelve el número de elementos en item (si un array o colección), o el número de caracteres (es un string).

EL extendido

Las funciones (2/2)

Función	Descripción
`fn:replace(string, before, after)`	Devuelve un string donde todas las ocurrencias del string before han sido reemplazadas por el string after.
`fn:split(string, separator)`	Devuelve un array donde los elementos son las partes del string separadas por separator.
`fn:startsWith(string, prefix)`	Devuelve true si el string comienza por prefix.
`fn:substring(string, begin, end)`	Devuelve la parte del string que comienza por el índice begin y que acaba en el índice end.
`fn:substringAfter(string, substring)`	Devuelve la parte del string que sigue a substring.
`fn:substringBefore(string, substring)`	Devuelve la parte de string que precede a substring.
`fn:toLowerCase(string)`	Devuelve un string con todos los caracteres de la entrada convertidos a minúsculas.
`fn:toUpperCase(string)`	Devuelve un string con todos los caracteres de la entrada convertidos a mayúsculas.
`fn:trim(string)`	Devuelve un string sin espacios en sus laterales.

Internacionalización I18N

Librería internacionalización

Acciones de formateo de números y fechas

- Cubre dos áreas:
  - ✓ Etiquetas (acciones) de formateo
  - ✓ Acciones de internacionalización

- Acciones de formateo:
  - ✓ Inspiradas en el funcionamiento de las clases **DateFormat** y **NumberFormat**
  - ✓ Para formatear un número usamos **formatNumber** con los atributos **number** para el número y **pattern** para el patrón de formateo a aplicar.

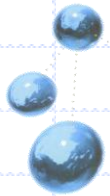
```
<fmt:formatNumber value="1000.001" pattern="#,#00.0#"/>
```

  - ✓ Si queremos parsear un número a partir de un string usamos **parseNumber**:

```
<fmt:parseNumber value="${currencyInput}" type="currency"
   var="parsedNumber"/>
```

  - ✓ Para formatear una fecha usamos **formatDate** y para parsear un string **parseDate**:

```
<jsp:useBean id="now" class="java.util.Date" />
<fmt:formatDate value="${now}" timeStyle="long"
   dateStyle="long"/>
<fmt:parseDate value="${dateInput}" pattern="MM dd, YYYY" />
```



Librería internacionalización

Acciones de Internacionalización

- Una pieza importante de la localización en Java es la clase `ResourceBundle`. Las acciones JSTL que permiten trabajar con esta clase son:
  - ✓ `fmt:bundle` para obtener un `ResourceBundle` correspondiente al Locale actual y
  - ✓ `fmt:message` para hacer lookups en el `ResourceBundle`
  - ✓ `fmt:parama` para agregar parámetros a un texto.

- El archivo Boundle:
  - ✓ Es un archivo de texto que se almacena junto con los recursos Java.
  - ✓ Se declara en el archivo web.xml de la siguiente manera:

```
<web-app id="WebApp_ID" version="2.4" …>
    ...
    <context-param>
     <param-name>
        javax.servlet.jsp.jstl.fmt.localizationContext
     </param-name>
     <param-value>
        cl.uchile.cc68j.taglibs.myboundle.properties
     </param-value>
    </context-param>
   ...
</web-app>
```

Librería internacionalización

Acciones de Internacionalización

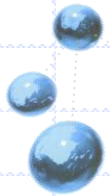
- El tag **fmt:message** permite desplegar el valor asociado a la clave que se pasa como parámetro.
- Esta clave debe estar en resource boundle. Ejemplo:

```
<fmt:bundle basename="myBundle">
    <fmt:message key="welcome"/>
</fmt:bundle>
```

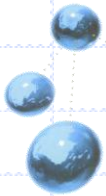
- Parámetros:
  - ✓ El tag **fmt:param**, anidado dentro de **fmt:message**, permite pasar parámetros a la llave.
  - ✓ Los parámetros en el boundle se usan con **{0}**, **{1}**, etc.
  - ✓ Ejemplo en boundle: **welcome=Bienvenido {0}!**.
  - ✓ Ejemplo de Uso:

```
<fmt:bundle basename="myBundle">
    <fmt:message key="welcome">
        <fmt:param value="Andres">
    <fmt:message key="welcome">
</fmt:bundle>
```

- Internacionalización:
  - ✓ Para internacionalizar los mensajes se crean nuevos archivos de boundle, con el mismo nombre del original, pero con una extensión asociada al locale: _fr_FR, _es_CL



Librería de etiquetas SQL



La librería de etiquetas SQL I

- JSTL permite una fácil integración con bases de datos
- No gestiona bien connection pooling, por tanto son solamente adecuadas para llevar a cabo prototipos o aplicaciones de bajo volumen.
- Ejemplo: seleccionar y visualizar un conjunto de elementos

```
<sql:setDataSource
      driver="com.cheapDrivers.jdbcDriver"
      url="jdbc:cheapDrivers:."
      user="guest"
      password="password"
      var="dataSource" />

<sql:query var="orderItems" dataSource="${dataSource}">
   SELECT * FROM items
   WHERE order_id = <cout value="${orderID}"/>
   ORDER BY price
</sql:query>
<table>
   <c:forEach var="row" items="${orderItems.rows}">
      <tr>
         <td><c:out value="${row.itemName}"/></td>
         <td><c:out value="${row.price}"/></td>
         <td><c:out value="${row.weight}"/></td>
      </tr>
   </c:forEach>
</table>
```

Creando la BBDD cc68j

```
CREATE DATABASE cc68j;

GRANT ALTER, SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
ON cc68j.*
TO cc68j@'%'
IDENTIFIED BY 'cc68j';

GRANT ALTER, SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
ON cc68j.*
TO cc68j@localhost
IDENTIFIED BY 'cc68j';

use cc68j;

CREATE TABLE EVENTOS(ID int(11) NOT NULL PRIMARY KEY,
NOMBRE VARCHAR(250), LOCALIZACION VARCHAR(250), FECHA bigint(20), DESCRIPCION
    VARCHAR(250));

INSERT INTO EVENTOS VALUES (0, 'SEMANA ESIDE', 'ESIDE-DEUSTO', 0, 'Charla sobre
    Python');
INSERT INTO EVENTOS VALUES (1, ‘CURSO J2EE', ‘U-CURSOS', 0, ‘Curso sobre tecnologías
    Java 2 Enterprise Edition');
```

Recuperando Datos de Tabla Eventos

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<sql:setDataSource
   driver="com.mysql.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/cc68j"
   user="cc68j"
   password="cc68j"
   var="dataSource"/>

<sql:query var="eventItems" dataSource="${dataSource}">
   select * from eventos
</sql:query>
<table>
   <c:forEach var="row" items="${eventItems.rows}">
      <tr>
         <td><c:out value="${row.ID}"/></td>
         <td><c:out value="${row.NOMBRE}"/></td>
         <td><c:out value="${row.LOCALIZACION}"/></td>
         <td><c:out value="${row.FECHA}"/></td>
         <td><c:out value="${row.DESCRIPCION}"/></td>
      </tr>
   </c:forEach>
</table>
```

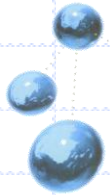
La librería de etiquetas SQL II

- También se soportan acciones para manejar transacciones (**sql:transaction**), **sql:update** soporta no sólo updates sino que también **insert** y **delete** e incluso **create**, es decir todas las acciones SQL que no devuelven un resultado:

```
<sql:transaction dataSource="${dataSource}">
    <sql:update>
    UPDATE account
    SET account_balance =account_balance -?
    WHERE accountNo = ?
    <sql:param value="${transferAmount}"/>
    <sql:param value="${sourceAccount}"/>
    </sql:update>
    <sql:update>
    UPDATE account
    SET account_balance =account_balance +?
    WHERE accountNo = ?
    <sql:param value="${transferAmount}"/>
    <sql:param value="${destAccount}"/>
    </sql:update>
</sql:transaction>
```

Recuperando Datos con JSTL de MySQL

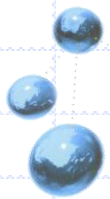
1. Instalar MySQL (http://www.mysql.com)
2. Instalar Connector/J, driver JDBC para MySQL (http://www.mysql.com/products/connector/j/)
3. Mover el driver MySQL a $CATALINA_HOME/common/lib.
4. Asegurarse que en $CATALINA_HOME/common/lib están las librerías de JSTL standard.jar y jstl.jar, si no bajarlas de: http://apache.rediris.es/jakarta/taglibs/standard/binaries/
5. Crear una tabla de prueba con el siguiente comando: mysql –uroot < createmysqldbtest.sql



Recuperando Datos con JSTL de MySQL

- Contenido de `createmysqldbtest.sql`:

```
GRANT ALL PRIVILEGES ON *.* TO javauser@localhost IDENTIFIED BY
      'javadude' WITH GRANT OPTION;
create database javatest;

use javatest;

create table testdata (
      id int not null auto_increment primary key,
      foo varchar(25),
      bar int
);

insert into testdata values(null, 'hello', 12345);
```



Recuperando Datos con JSTL de MySQL

5. Añadir el siguiente párrafo a $CATALINA_HOME/conf/server.xml:

```
<Context path="/DBTest" docBase="DBTest"
        debug="5" reloadable="true" crossContext="true">
  <Resource name="jdbc/TestDB" auth="Container"
      type="javax.sql.DataSource"
                maxActive="100" maxIdle="30" maxWait="10000"
                username="javauser" password="javadude"
      driverClassName="com.mysql.jdbc.Driver"

      url="jdbc:mysql://localhost:3306/javatest?autoReconnect=true"/
      >
</Context>
```

Recuperando Datos con JSTL de MySQL

6. Crear un fichero de configuración para esta aplicación `WEB-INF/web.xml`:

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
  <description>MySQL Test App</description>
  <resource-ref>
      <description>DB Connection</description>
      <res-ref-name>jdbc/TestDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

Recuperando Datos con JSTL de MySQL

7. Crear un fichero `test.jsp`:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<sql:query var="rs" dataSource="jdbc/TestDB">
select id, foo, bar from testdata
</sql:query>

<html>
  <head>
    <title>DB Test</title>
  </head>
  <body>

  <h2>Results</h2>

<c:forEach var="row" items="${rs.rows}">
    Foo ${row.foo}<br/>
    Bar ${row.bar}<br/>
</c:forEach>

  </body>
</html>
```