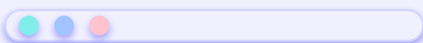




JSTL

¿Qué es?



JSTL es un conjunto de librerías de etiquetas simples y estándares que encapsulan la funcionalidad principal que es usada comúnmente para escribir páginas JSP.

Contiene muchas etiquetas personalizadas JSP comunes y útiles.

JSTL provee un conjunto de cinco librerías estándar:

- **Core**
- **Internationalization/format**
- **XML**
- **SQL**



Una etiqueta JSTL corresponde a una acción, llamándolas acción nos indica que añaden comportamiento dinámico a una página estática.

EL (Lenguaje de expresiones)

El lenguaje de expresiones “EL” simplemente define un mecanismo para expresar expresiones simples en una sintaxis muy sencilla.

En EL las expresiones están delimitadas por `${}`.

Sintaxis: `${expresión}`, expresión es el valor presente que se evalúa en tiempo de ejecución y se envía al flujo de salida.

Ejemplo

Para imprimir el valor del parámetro de solicitud uname:

```
<% uname=request.getParameter("Ana");%>
```

En este ejemplo, podemos recuperar un parámetro de solicitud en particular sin usar código Java usando la sintaxis del lenguaje de expresión. `${param.uname}`

Objetos implícitos en lenguaje de expresión JSP

Hay muchos objetos implícitos en el lenguaje de expresión. Son los siguientes:

Objetos implícitos	Uso
pageScope	asigna el nombre de atributo dado con el valor establecido en el alcance de la página
requestScope	asigna el nombre de atributo dado con el valor establecido en el alcance de la solicitud
sessionScope	asigna el nombre de atributo dado con el valor establecido en el alcance de la sesión
applicationScope	asigna el nombre de atributo dado con el valor establecido en el ámbito de la aplicación
param	asigna el parámetro de solicitud al valor único
paramValues	asigna el parámetro de solicitud a una matriz de valores
header	asigna el nombre del encabezado de la solicitud al valor único
headerValues	asigna el nombre del encabezado de la solicitud a una matriz de valores
cookie	asigna el nombre de la cookie dada al valor de la cookie
initParam	mapea el parámetro de inicialización
pageContext	proporciona acceso a muchas solicitudes de objetos, sesiones, etc.

Operadores EL

El soporte de expresiones EL proporcionado por la mayoría de los operadores aritméticos y lógicos de Java:

operadores	descripción
.	Frijol acceso a una propiedad o una entrada de asignación
[]	Acceder a una matriz o lista de elementos
()	Organizar una sub-expresión para cambiar la prioridad
+	más
-	Guardar o negativo
*	Multiplicar
/ Div O	excepto
O mod%	Modulo
== O eq	Prueba para la igualdad
!= O ne	Probar si la desigualdad
<O lt	Probar si menos de
> O gt	La prueba es mayor que
<= O le	Probar si menor o igual
> = O ge	Probar si mayor o igual
&& O y	la lógica de prueba y
O o	la lógica de prueba o
! O no	prueba negada
vacía	Probar si nula

Funciones en lenguaje de expresiones

El lenguaje de expresiones también permite usar funciones en expresiones. Estas se definen en las bibliotecas de etiquetas personalizadas. El uso de una función tiene la siguiente sintaxis:

```
${ns:func(param1, param2, ...)}
```

Donde ns es el espacio de nombres de la función, func es el nombre de la función y param1 es el primer valor del parámetro.

```
${fn:length("Get my length")}
```




Librería Core





¿Qué es?

Es una librería que implementa acciones de propósito general, como mostrar información, crear y modificar variables de distinto ámbito y tratar excepciones.





La librería core incluyen etiquetas para:

- Funciones de propósito general.
- Funciones de control de flujo.
- Funciones de acceso a URLs.

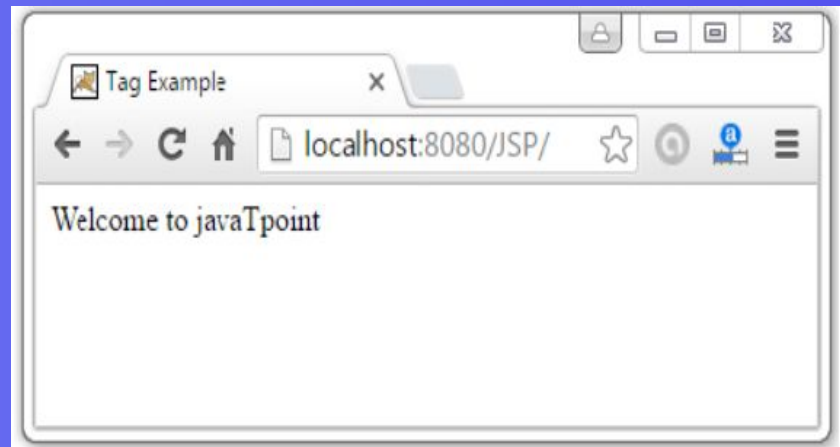
Tags de propósito general

- 
- **out**
 - **set**
 - **remove**
 - **catch**

out

Muestra el resultado de una expresión, similar a la forma en que funcionan las etiquetas `<%=...%>`.

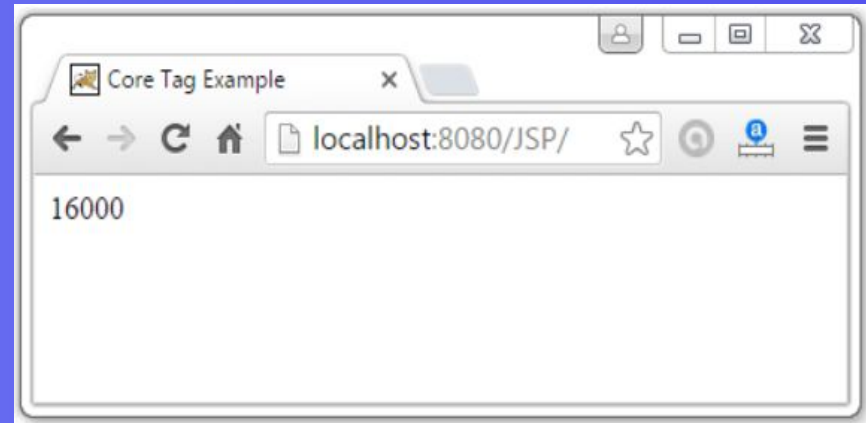
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Tag Example</title>
</head>
<body>
<c:out value="${Welcome to javaTpoint}"/>
</body>
</html>
```



set

La etiqueta `<c:set>` es útil porque evalúa la expresión y usa el resultado para establecer un valor de `java.util.Map` o `JavaBean`.

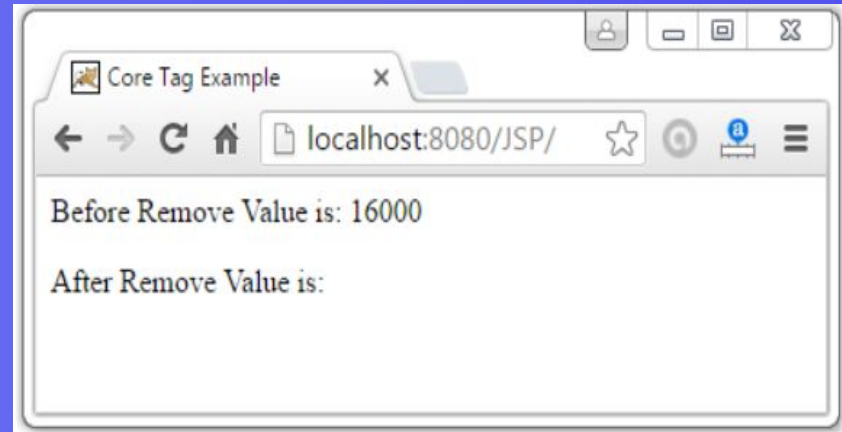
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="Income" scope="session" value="${4000*4}"/>
<c:out value="${Income}"/>
</body>
</html>
```



remove

La etiqueta `<c:remove>` elimina la variable de un primer ámbito o de un ámbito específico, es similar a la etiqueta de acción `jsp:setProperty`.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<p>Before Remove Value is: <c:out value="${income}"/> </p>
<c:remove var="income"/>
<p>After Remove Value is: <c:out value="${income}"/> </p>
</body>
</html>
```



catch

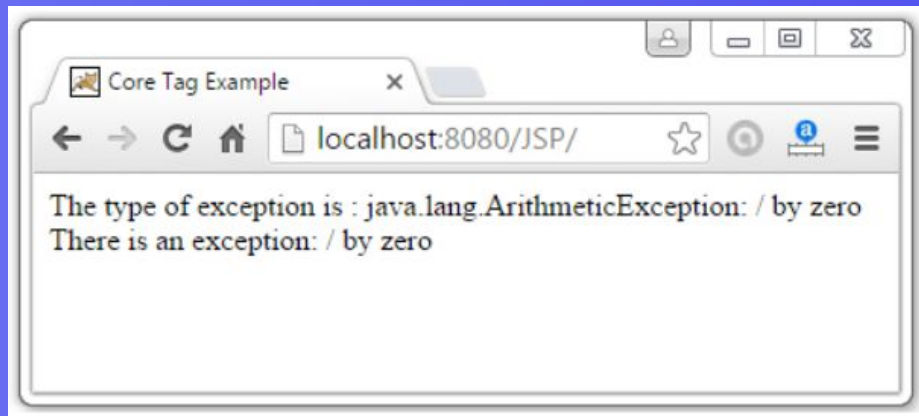
La etiqueta `< c:catch >` captura cualquier excepción Throwable que ocurra en el cuerpo y, opcionalmente, lo expone.

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


<c:catch var="catchtheException">
  <% int x = 2/0;%>
</c:catch>

<c:if test = "${catchtheException != null}">
  <p>The type of exception is : ${catchtheException} <br />
  There is an exception: ${catchtheException.message}</p>
</c:if>

</body>
</html>
```



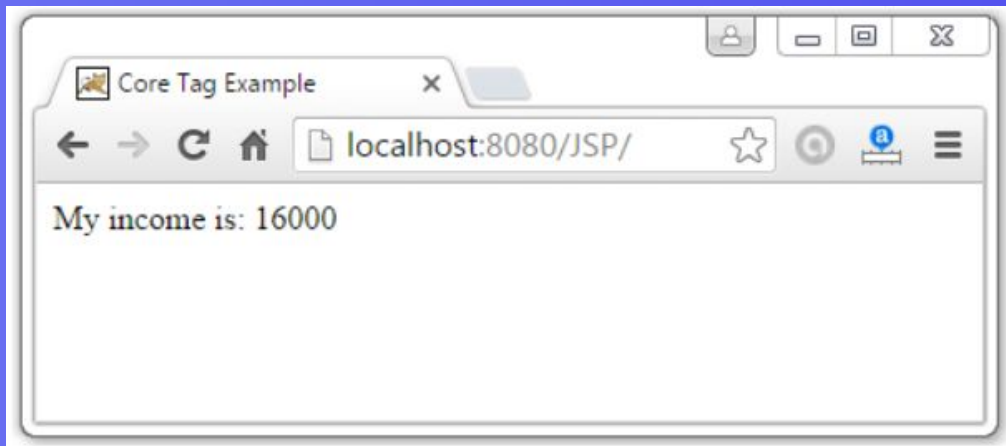
Tags de control de flujo

- 
- **if**
 - **choose, When y otherwise**
 - **forEach**
 - **forTokens**

if

La etiqueta `<c:if>` se usa para probar la condición y muestra el contenido del cuerpo, si la expresión evaluada es verdadera.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<c:if test="${income > 8000}">
  <p>My income is: <c:out value="${income}"/></p>
</c:if>
</body>
</html>
```





choose, When y otherwise



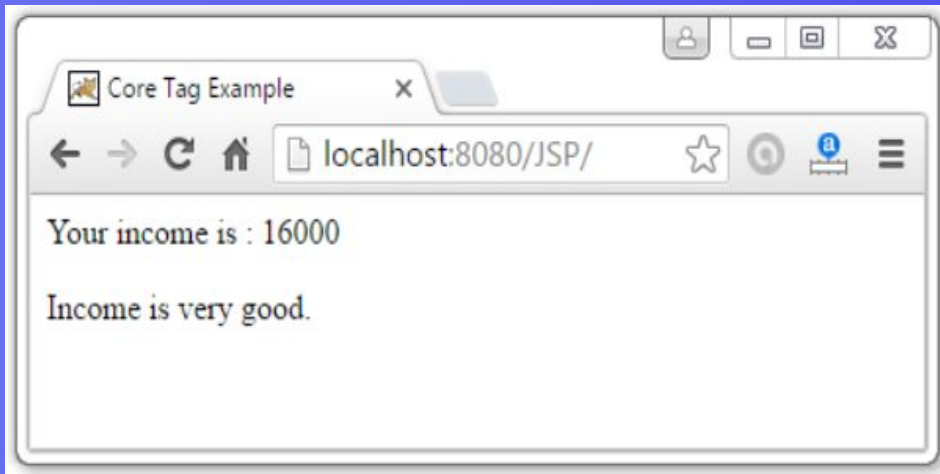
La etiqueta `< c:choose >` es una etiqueta condicional que establece un contexto para operaciones condicionales mutuamente excluyentes.

`<c:when>` es una subetiqueta de `<choose>` que incluirá su cuerpo si la condición evaluada es 'verdadera'.

- `<c:otherwise>` también es una subetiqueta de `< choose >`, sigue
- a las etiquetas `<when>` y se ejecuta sólo si todas las
- condiciones previas evaluadas son 'falsas'.
-

Ejemplo

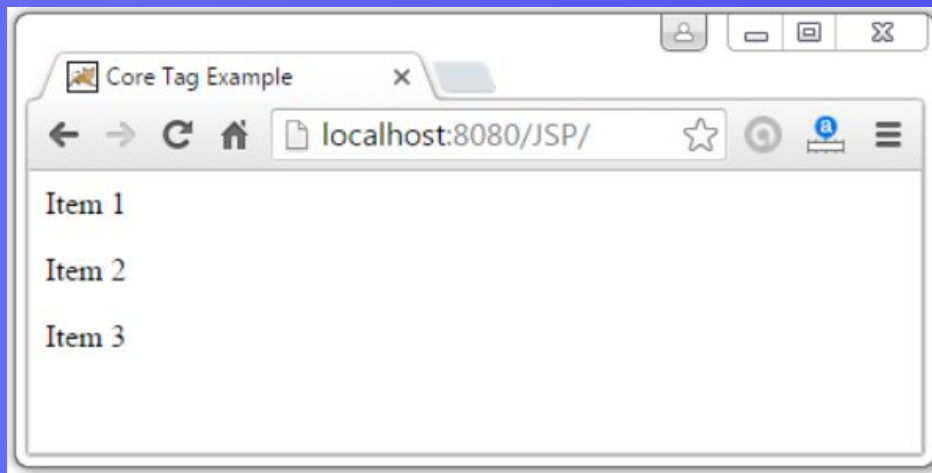
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="{4000*4}"/>
<p>Your income is : <c:out value="{income}"/> </p>
<c:choose>
  <c:when test="{income <= 1000}">
    Income is not good.
  </c:when>
  <c:when test="{income > 10000}">
    Income is very good.
  </c:when>
  <c:otherwise>
    Income is undetermined...
  </c:otherwise>
</c:choose>
</body>
</html>
```



forEach

`<c:forEach>` es una etiqueta de iteración que se usa para repetir el contenido del cuerpo anidado un número fijo de veces o sobre la colección.

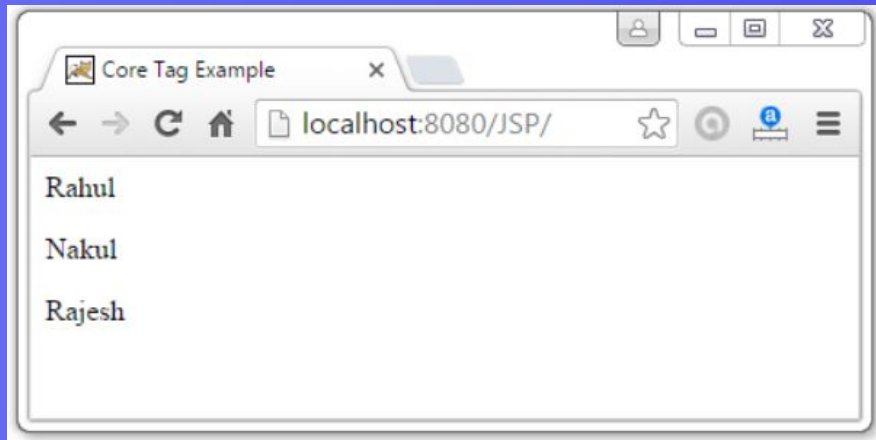
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:forEach var="j" begin="1" end="3">
  Item <c:out value="{j}"/><p>
</c:forEach>
</body>
</html>
```



forTokens

La etiqueta `< c:forTokens >` itera sobre tokens separados por los delimitadores proporcionados. Se utiliza para dividir una cadena en tokens e iterar a través de cada uno de los tokens para generar resultados.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:forTokens items="Rahul-Nakul-Rajesh" delims="-" var="name">
  <c:out value="${name}"/> <p>
</c:forTokens>
</body>
</html>
```



Tags de manejo de URLs

- 
- **import**
 - **param**
 - **url**

Import

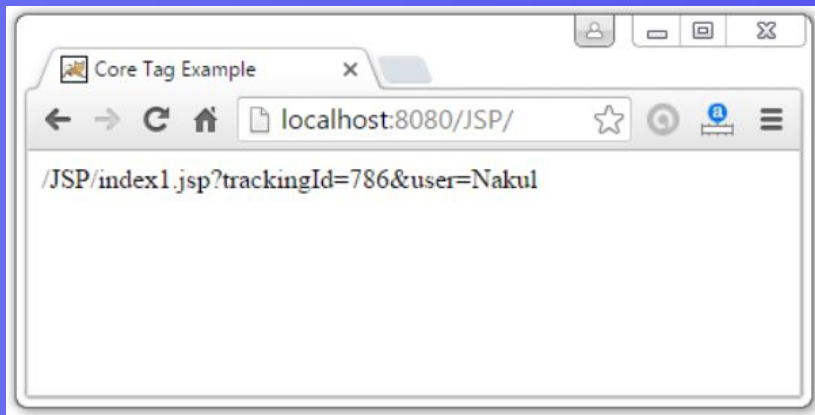
`<c:import>` es similar a 'include' de jsp, con una característica adicional de incluir el contenido de cualquier recurso dentro o fuera del servidor.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Tag Example</title>
</head>
<body>
<c:import var="data" url="http://www.javatpoint.com"/>
<c:out value="${data}"/>
</body>
</html>
```


Param

La etiqueta `< c:param >` agrega el parámetro en una URL de etiqueta de 'importación' que lo contiene.

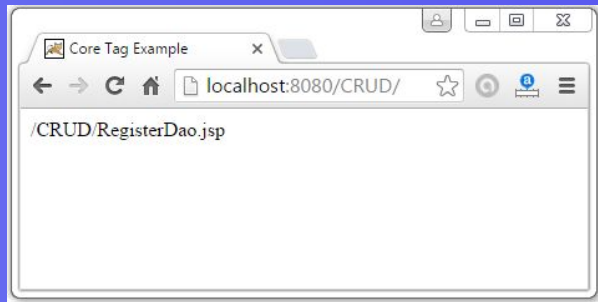
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:url value="/index1.jsp" var="completeURL"/>
<c:param name="trackingId" value="786"/>
<c:param name="user" value="Nakul"/>
</c:url>
${completeURL}
</body>
</html>
```



url

La etiqueta `< c:url >` crea una URL con un parámetro de consulta opcional. Esta etiqueta realiza automáticamente la operación de reescritura de URL.

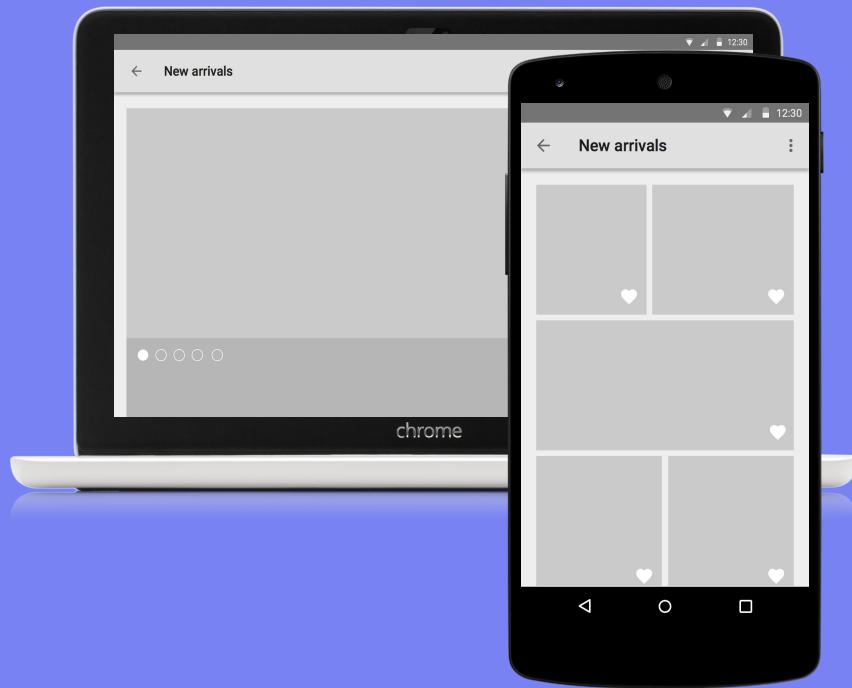
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:url value="/RegisterDao.jsp"/>
</body>
</html>
```



Librería formatting

Biblioteca de etiquetas formateadas JSTL: para usar la biblioteca de etiquetas formateadas JSTL, primero debe importar la biblioteca de etiquetas a través de la instrucción taglib en la página.

```
<%@ taglib uri=" prefix="fmt" %>
```





Etiqueta formatDate

Permite formatear fechas.

```
<fmt:formatDate value="${pageScope.fecha}" pattern="dd/MM/yyyy"/>
```

Etiqueta parseDate

Permite convertir una cadena en una fecha, hora o fecha y hora

```
<c:set var="fecha2" value="01/01/2013" />
```

```
<fmt:parseDate value="${fecha2}" var="fechaParse" pattern="dd/MM/yyyy" />
```



Etiqueta formatNumber

Permite dar formato a valores numéricos.

```
<c:set var="numero">123</c:set>  
<br /><fmt:formatNumber value="{numero}" pattern="00000000"/>
```

Etiqueta parseNumber

Permite convertir a número, porcentaje o moneda.

```
<c:set var="num" value="99.99" />  
num = <fmt:formatNumber type="number" value="{num}" />  
<br />  
<fmt:parseNumber type="number" var="num1" value="{num}" integerOnly="true" pattern="#.##" />  
num1 = <fmt:formatNumber type="number" value="{num1}" />
```



Etiqueta bundle

Carga un archivo de recursos, para que sea utilizado por las etiquetas message.

Etiqueta message

Muestra un mensaje internacionalizado.

```
<fmt:bundle basename="com.me.jsp.bundle.num" prefix="color.">
  <fmt:message key="red"/><br />
  <fmt:message key="green"/><br />
  <fmt:message key="blue"/><br />
</fmt:bundle>
```

Etiqueta timeZone

Especifica la zona horaria para cualquier etiqueta de conversión o formateo que se encuentre en su cuerpo de timeZone.

```
<fmt:timeZone value="${tz}">
  <fmt:formatDate value="${pageScope.fecha}"
    timeZone="${tz}" type="both" />
</fmt:timeZone>
```