

## JSTL

### FMT Tag Library

Las etiquetas de la librería FMT están divididas en cuatro grupos. El primer grupo permite establecer la internacionalización del resto de las etiquetas que se utilicen. En otras palabras, permite establecer explícitamente la configuración regional y la zona horaria que las etiquetas FMT utilizarán para dar formato. El segundo y tercer grupo permiten dar formato a las fechas y números, respectivamente. Y el último grupo, permite internacionalizar los mensajes de texto.

#### Internacionalización

La configuración regional utilizada por las etiquetas JSTL, se determina normalmente mediante la evaluación del atributo "Accept-Language" de la cabecera HTTP enviada como respuesta a la petición del usuario. Si esta información no está presente, entonces JSTL proporciona un conjunto de variables de configuración JSP para configurar un idioma predeterminado. Si a su vez, estas variables de configuración no han sido establecidas, la máquina virtual toma la configuración regional por defecto del Sistema Operativo donde se está ejecutando el contenedor JSP.

La etiqueta JSTL utilizada para tal fin es `<fmt:setLocale>` cuya sintaxis es:

**`<fmt:setLocale value="expression" scope="scope" variant="expression"/>`**

value: Cadena con el nombre de la localización seleccionada o una instancia de la clase `java.util.Locale`. El nombre está construido por el código ISO del país en minúscula seguido opcionalmente por un guión más el código ISO del lenguaje en mayúscula. Este atributo es el único obligatorio.

scope: Ámbito de visibilidad de la localización definida. Los valores posibles son los ya estudiados anteriormente.

variant: Permite personalizar la configuración regional a un navegador específico. Por ejemplo, Mac y Win son variantes de los nombres Apple Macintosh y Microsoft Windows, respectivamente.

Después que el contenedor JSP procesa este fragmento, las preferencias del idioma definidas por el navegador serán ignoradas.

La etiqueta `<fmt:setTimeZone>` permite fijar la zona horaria predeterminada para el uso de las otras etiquetas FMT. Su sintaxis es:

**`<fmt:setTimeZone value="expression" var="name" scope="scope"/>`**

value: Cadena con el nombre de la zona horaria seleccionada o una instancia de la clase `java.util.TimeZone`. Desafortunadamente, no hay norma para la definición del huso horario, por lo que se utilizan los definidos por la plataforma Java, los valores válidos se pueden tomar de la ejecución del método `getAvailableIDs()`. Este atributo es el único obligatorio.

var: Nombre de la variable que referencia el huso horario que se declara.

scope: Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

También se puede utilizar la etiqueta `<fmt:timeZone>` donde la aplicación del huso horario es válida dentro del cuerpo de la etiqueta. Su sintaxis es:

**`<fmt:timeZone value="expression">`  
**TimeZone-Body**  
**`</fmt:timeZone>`****

#### Fechas

La librería FMT incluye dos etiquetas para interactuar con la fecha y hora: `<fmt:formatDate>` y `<fmt:parseDate>`. Como sus nombres lo indican, se utiliza `<fmt:formatDate>` para dar formato y mostrar la fecha y hora, mientras que `<fmt:parseDate>` se utiliza para analizar la fecha y los valores de la hora.

La sintaxis de la etiqueta `<fmt:formatDate>` es:

**`<fmt:formatDate value="expression" timeZone="expression" type="field" dateStyle="style" timeStyle="style" pattern="expression" var="name" scope="scope"/>`**

value: Es una instancia de la clase `java.util.Date`. Este atributo es el único obligatorio.

**timeZone:** Permite definir bajo que huso horario se mostrará la fecha y hora especificada. Si no es definida se toma la utilizada por la máquina virtual, es decir, la configuración horaria del Sistema Operativo.

**type:** Indica que campos de la fecha se mostrarán. Los valores posibles son: "time", "date" o "both". El valor por defecto es "date".

**dateStyle:** Permite definir el formato utilizado en la fecha. Los valores posibles son: "default", "short", "medium", "long", y "full". El valor por defecto es "default".

**timeStyle:** Permite definir el formato utilizado en la hora. Los valores posibles son: "default", "short", "medium", "long", y "full". El valor por defecto es "default".

**pattern:** En lugar de definir un estilo, se puede utilizar un patrón. El patrón se basa en las convenciones de la clase java.text.SimpleDateFormat.

**var:** Nombre de la variable que referencia al formato que se declara.

**scope:** Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

Veamos un ejemplo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo</title>
  </head>
  <body>
    <h2>Formatos de Fechas</h2>
    <jsp:useBean id="hoy" class="java.util.Date" />
    <fmt:setLocale value="en_US" />
    <fmt:setTimeZone value="GMT" var="gmt" />
    <p>Fecha completa (en_US): <fmt:formatDate value="{hoy}" dateStyle="full" /></p>
    <p>Fecha con formato MM/d/yyyy hh:mm (en_US): <fmt:formatDate value="{hoy}"
timeZone="{gmt}" dateStyle="full" timeStyle="medium" pattern="MM/d/yyyy hh:mm"/></p>
    <br>
    <fmt:setLocale value="es_AR" />
    <p>Fecha completa (es_AR): <fmt:formatDate value="{hoy}" dateStyle="full" /></p>
    <p>Fecha con formato d/MM/yyyy hh:mm (es_AR): <fmt:formatDate value="{hoy}"
dateStyle="full" timeStyle="medium" pattern="d/MM/yyyy hh:mm"/></p>
  </body>
</html>
```

La etiqueta <fmt:parseDate> presenta dos posibles sintaxis:

Sintaxis 1

```
<fmt:parseDate value="expression" type="field" dateStyle="style" timeStyle="style"
pattern="expression" timeZone="expression" parseLocale="expression" var="name" scope="scope"/>
```

Sintaxis 2

```
<fmt:parseDate type="field" dateStyle="style" timeStyle="style" pattern="expression"
timeZone="expression" parseLocale="expression" var="name" scope="scope">
  ParseDate-Body
</fmt:parseDate>
```

**value:** Cadena de caracteres que puede ser una fecha, hora o ambas. Este atributo es el único obligatorio.

**timeZone:** Permite definir bajo que huso horario se mostrará la fecha y hora especificada. Si no es definida se toma la utilizada por la máquina virtual, es decir, la configuración horaria del Sistema Operativo.

**type:** Indica que campos de la fecha se mostrarán. Los valores posibles son: "time", "date" o "both". El valor por defecto es "date".

dateStyle: Permite definir el formato utilizado en la fecha. Los valores posibles son: "default", "short", "medium", "long", y "full". El valor por defecto es "default".

timeStyle: Permite definir el formato utilizado en la hora. Los valores posibles son: "default", "short", "medium", "long", y "full". El valor por defecto es "default".

pattern: En lugar de definir un estilo, se puede utilizar un patrón. El patrón se basa en las convenciones de la clase java.text.SimpleDateFormat.

parseLocale: Permite definir la configuración regional que se utilizará. Debe ser una cadena con el nombre de la localización seleccionada o una instancia de la clase java.util.Locale.

var: Nombre de la variable que referencia al objeto Date creado.

scope: Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

ParseDate-Body: Cuando se define no se debe utilizar el atributo "value", el valor especificado debe ser según la definición de este último.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo</title>
  </head>
  <body>
    <h2>Parseo de Fecha</h2>
    <c:set var="fechaNac" value="14/10/1980 17:12:00" />
    <c:catch var="ex">
      <fmt:parseDate parseLocale="es_AR" type="both" dateStyle="short" timeStyle="short"
var="fechaNacP">
        <fmt:formatDate value="${fechaNacP}" dateStyle="short" timeStyle="short" type="both"
/></p>
      </p>
    </c:catch>
    <c:if test="${!empty ex}">
      <p>La fecha informada no es válida. Error: ${ex}</p>
    </c:if>
  </body>
</html>
```

### Números

Así como las etiquetas <fmt:formatDate> y <fmt:parseDate> son utilizadas para dar formato y analizar fechas, las etiquetas <fmt:formatNumber> y <fmt:parseNumber> realizan las mismas funciones sobre los datos numéricos.

La etiqueta <fmt:formatNumber> se utiliza para mostrar datos numéricos, incluyendo las monedas y los porcentajes, de acuerdo a la configuración regional especificada. Su sintaxis es:

```
<fmt:formatNumber value="expression" type="type" pattern="expression" currencyCode="expression"
currencySymbol="expression" maxIntegerDigits="expression" minIntegerDigits="expression"
maxFractionDigits="expression" minFractionDigits="expression" groupingUsed="expression"
var="name" scope="scope"/>
```

value: Número al que se dará formato. Este atributo es el único obligatorio.

type: Indica el tipo de formato con el que se tratará el número. Los valores posibles son: "number", "currency" o "percent". El valor por defecto es "number".

pattern: Este atributo tiene prioridad sobre "type". Es un patrón basado en las convenciones de la clase `java.text.DecimalFormat`.

currencyCode: Cuando el atributo "type" tiene como valor "currency", se pueden utilizar este atributo para especificar explícitamente la moneda a utilizar. Los valores posibles se basan en la norma ISO de definición de monedas. Puede ser una instancia de la clase `java.util.Currency`.

currencySymbol: Sirve para explicitar el símbolo de la moneda. El atributo `currencyCode` tiene prioridad sobre este atributo.

maxIntegerDigits, minIntegerDigits, maxFractionDigits y minFractionDigits: Estos atributos se utilizan para controlar el número de dígitos desplegados antes y después del punto decimal. El valor a especificar debe ser número entero.


groupingUsed: Es un valor booleano y controla si los dígitos antes del punto decimal se agrupan. Su valor por defecto es "true".

var: Nombre de la variable que referencia al formato que se declara.

scope: Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

Veamos un ejemplo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Ejemplo</title>
</head>
<body>
<h2>Formato de Números</h2>
<h3>Número 712.03</h3>
<c:set var="val1" value="712.03" />
<fmt:setLocale value="en_US"/>
<p>Formato (en_US): <fmt:formatNumber value="{val1}" /></p>
<fmt:setLocale value="es_AR"/>
<p>Formato (es_AR): <fmt:formatNumber value="{val1}" /></p>
<fmt:setLocale value="de_DE"/>
<p>Formato (de_DE): <fmt:formatNumber value="{val1}" /></p>
<br>
<h3>Porcentaje 63.5</h3>
<c:set var="val2" value="63.5" />
<fmt:setLocale value="en_US"/>
<p>Porcentaje (en_US): <fmt:formatNumber value="{val2}" type="percent" /></p>
<fmt:setLocale value="es_AR"/>
<p>Porcentaje (es_AR): <fmt:formatNumber value="{val2}" type="percent" /></p>
<fmt:setLocale value="de_DE"/>
<p>Porcentaje (de_DE): <fmt:formatNumber value="{val2}" type="percent" /></p>
<br>
<h3>Moneda 631.09</h3>
<c:set var="val3" value="631.09" />
<fmt:setLocale value="en_US"/>
<p>Moneda (en_US): <fmt:formatNumber value="{val3}" type="currency" /></p>
<fmt:setLocale value="es_AR"/>
<p>Moneda (es_AR): <fmt:formatNumber value="{val3}" type="currency" /></p>
<fmt:setLocale value="de_DE"/>
<p>Moneda (de_DE): <fmt:formatNumber value="{val3}" type="currency" /></p>
<h3>Uso de un patrón 123456789</h3>
<c:set var="val4" value="123456789" />
<p>###.###E0: <fmt:formatNumber value="{val4}" pattern="###.###E0" /></p>
</body>
</html>
```

	INGENIERÍA EN INFORMÁTICA – PLAN 2003 DISEÑO AVANZADO SOFTWARE – 10º CUATRIMESTRE	
	APUNTE SOBRE JSTL FMT	VERSIÓN: 1.3 VIGENCIA: 06-08-2009

La etiqueta `<fmt:parseNumber>` permite analizar un número proporcionado a través del atributo "value" o en el cuerpo de la etiqueta según la configuración regional especificada. Su resultado es una instancia de la clase `java.lang.Number`. Presenta dos posibles sintaxis:

Sintaxis 1

```
<fmt:parseNumber value="expression" type="type" pattern="expression" parseLocale="expression" integerOnly="expression" var="name" scope="scope" />
```

Sintaxis 2

```
<fmt:parseNumber type="type" pattern="expression" parseLocale="expression" integerOnly="expression" var="name" scope="scope">  
ParseNumber-Body  
</fmt:parseNumber>
```

value: Número que se analizará. Este atributo es el único obligatorio.

type: Indica cómo se tratará al número. Los valores posibles son: "number", "currency" o "percent". El valor por defecto es "number".

pattern: Este atributo tiene prioridad sobre "type". Es un patrón basado en las convenciones de la clase `java.text.DecimalFormat`.

parseLocale: Permite definir la configuración regional que se utilizará. Debe ser una cadena con el nombre de la localización seleccionada o una instancia de la clase `java.util.Locale`.

integerOnly: Es un valor booleano e indica si solo debe analizarse la parte entera del número, ignorando los dígitos decimales. Su valor por defecto es "false".

var: Nombre de la variable que referencia al objeto Number creado.


scope: Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

ParseNumber-Body: Cuando se define no se debe utilizar el atributo "value", el valor especificado debe ser según la definición de este último.

Veamos un ejemplo sencillo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>  
<!DOCTYPE html>  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>Ejemplo</title>  
  </head>  
  <body>  
    <h2>Parseo de Números</h2>  
    <h3>Parseo de moneda</h3>  
    <fmt:parseNumber var="val1" type="currency" value="$123456,789" parseLocale="es_AR" />  
    <p>${val1}</p>  
    <br>  
    <h3>Parseo de enteros</h3>  
    <fmt:parseNumber var="val2" value="123456,789" integerOnly="true" parseLocale="es_AR" />  
    <p>${val2}</p>  
    <br>  
    <h3>Parseo de números</h3>  
    <fmt:parseNumber var="val3" value="123456,789" parseLocale="es_AR" />  
    <p>${val3}</p>  
    <br>  
    <h3>Parseo de porcentajes</h3>  
    <fmt:parseNumber var="val4" type="percent" value="123456,789%" parseLocale="es_AR" />  
    <p>${val4}</p>  
  </body>  
</html>
```

Texto

	INGENIERÍA EN INFORMÁTICA – PLAN 2003 DISEÑO AVANZADO SOFTWARE – 10º CUATRIMESTRE	
	APUNTE SOBRE JSTL FMT	VERSIÓN: 1.3 VIGENCIA: 06-08-2009

La posibilidad de internacionalizar texto se logra a través de la etiqueta `<fmt:message>`. Esta etiqueta permite recuperar mensajes de texto desde un paquete de recursos teniendo en cuenta la configuración regional especificada para mostrarlo en una página JSP.

Los paquetes de recursos para almacenar los mensajes de acuerdo a una configuración regional pueden ser una clase o archivo de propiedades cuyo nombre se adhiere a una determinada nomenclatura, en la que se combina un nombre de archivo más la configuración regional. Por ejemplo, podríamos tener el archivo `MisMensajes_es.properties` para mensajes en español y `MisMensajes_en.properties` para mensajes en inglés. Hasta llegado el caso se puede especificar para un determinado país, `MisMensajes_es_AR.properties`.

En cada uno de estos archivos se definen las mismas propiedades pero los valores de estas propiedades se pueden personalizar por idioma o dialecto según corresponda. A cada propiedad se le pueden especificar parámetros que se definen secuencialmente comenzando con cero y encerrando el número entre llaves.

Por ejemplo,

`MisMensajes_es.properties`

**`saludo=iHola {0}!!!`**

**`mensaje=iQué tenga un buen día!!! :)`**

`MisMensajes_en.properties`

**`saludo=Hi {0}!!!`**

**`mensaje=Have a nice day!!! :)`**

Nota: Estos archivos serán almacenados en un paquete asociado al proyecto con el nombre "properties".

El primer paso para mostrar dicho contenido es especificar el paquete de recursos. La librería FMT posee dos etiquetas para lograr esto `<fmt:setBundle>` y `<fmt:bundle>`.

La etiqueta `<fmt:setBundle>` establece un paquete de recursos por defecto para ser utilizados por la etiqueta `<fmt:message>` dentro de un alcance determinado. Mientras que la etiqueta `<fmt:bundle>` establece el paquete de recursos para ser usado dentro del cuerpo de la etiqueta, por lo que las etiquetas `<fmt:message>` se encuentran anidadas dentro de esta.

**`<fmt:setBundle basename="expression" var="name" scope="scope"/>`**

basename: Atributo obligatorio que permite identificar el paquete de recursos a utilizar. Debe tenerse en cuenta que el nombre a especificar no debe contener la configuración regional, es decir, en nuestro ejemplo se debe especificar solamente "properties.MisMensajes"

var: Nombre de la variable que referencia al paquete de recursos.

scope: Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente.

Por su parte la etiqueta `<fmt:bundle>` presenta la siguiente sintaxis:

**`<fmt:bundle basename="expression" prefix="expression">`**

**`Bundle-Body`**

**`</fmt:bundle>`**

basename: Atributo obligatorio que permite identificar el paquete de recursos a utilizar. Debe tenerse en cuenta que el nombre a especificar no debe contener la configuración regional, es decir, en nuestro ejemplo se debe especificar solamente "properties.MisMensajes"

prefix: Atributo opcional que permite definir un prefijo para utilizar en las etiquetas `<fmt:message>` anidadas.

Una vez que se ha definido el paquete de recursos a utilizar, las propiedades se acceden con la etiqueta `<fmt:message>`. La misma presenta dos sintaxis posibles:

Sintaxis 1

**`<fmt:message key="expression" bundle="expression" var="name" scope="scope"/>`**

Sintaxis 2

**`<fmt:message key="expression" bundle="expression" var="name" scope="scope">`**

**`<fmt:param value="expression"/>`**

**`...`**

**`</fmt:message>`**

key: Atributo obligatorio que permite especificar la propiedad del paquete de recursos a mostrar.

**bundle:** Sirve para definir un paquete de recursos específico en donde buscar el mensaje asociado a la propiedad indicada en el atributo "key". Su valor debe ser la variable definida a través de la etiqueta <fmt:setBundle> o el prefijo de la etiqueta <fmt:bundle>.

**var:** Nombre de la variable que contiene el mensaje de texto generado por la etiqueta. Al usar este atributo no se imprime el mensaje dentro de la página JSP.

**scope:** Ámbito de visibilidad de la variable definida. Los valores posibles son los ya estudiados anteriormente. Se utiliza la etiqueta <fmt:param> para proporcionar los valores de los parámetros del mensaje de texto. Cabe destacar que es importante el orden en que se define cada una de estas etiquetas dentro de <fmt:message>.

Veamos un ejemplo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo</title>
  </head>
  <body>
    <h2>Internacionalización</h2>
    <fmt:setLocale value="en_US" />
    <fmt:setBundle basename="ar.edu.ubp.das.properties.misMensajes" var="etq" />
    <h4>
      <fmt:message key="saludo" bundle="${etq}">
        <fmt:param value="Mariela" />
      </fmt:message>
      <fmt:message key="mensaje" bundle="${etq}" />
    </h4>
    <fmt:setLocale value="es_AR" />
    <fmt:bundle basename="ar.edu.ubp.das.properties.misMensajes">
      <h4>
        <fmt:message key="saludo">
          <fmt:param value="Mariela" />
        </fmt:message>
        <fmt:message key="mensaje" />
      </h4>
    </fmt:bundle>
  </body>
</html>
```