

JSTL

XML Tag Library

XML proporciona un medio flexible para representar datos estructurados. Como resultado de ello, es especialmente adecuado para el intercambio de datos entre sistemas débilmente acoplados. Esto a su vez hace que sea una tecnología de integración atractiva para aplicaciones Web.

El primer paso para interactuar con los datos XML es recuperarlo como un documento XML y armar una estructura de datos para acceder al contenido del mismo. Después de que el documento ha sido analizado se puede, opcionalmente, transformar para producir un nuevo documento XML, al que se le pueden aplicar las mismas operaciones. Finalmente, los datos del documento se pueden extraer y mostrar o utilizar como entrada para realizar operaciones adicionales.

Estas tareas se reflejan en las etiquetas JSTL utilizadas para la manipulación de XML. Los documentos XML se recuperan mediante la etiqueta `<c:import>` de la librería "Core". La etiqueta `<x:parse>` se utiliza para analizar el documento, con soporte para las tecnologías estándar de análisis XML como son el Modelo de Objetos de Documento (DOM) y la API Simple para XML (SAX). La etiqueta `<x:transform>` está disponible para la transformación de documentos XML y se basa en la tecnología estándar para la transformación de datos XML, Extensible Stylesheet Language (XSL). Por último, varias etiquetas se proporcionan para acceder y manipular datos XML, los cuales utilizan un nuevo estándar, el lenguaje XML Path (XPath).

A su vez, la librería posee otras etiquetas muy similares a las "etiquetas fundamentales". Por ejemplo, al igual que `<c:out>` tenemos `<x:out>`. Del mismo modo, `<x:forEach>`, `<x:if>`, `<x:when>`, etc. Por lo tanto, si hemos entendido la sintaxis de la librería "Core", las etiquetas XML no serán difíciles de utilizar.

Parseo de XML

Sintaxis 1

```
<x:parse xml="xml" var="name" scope="scope" filter="expression" systemId="expression"/>
```

Sintaxis 2

```
<x:parse var="name" scope="scope" filter="expression" systemId="expression">  
  xml  
</x:parse>
```

xml: Su valor debe ser una cadena que contiene el documento XML para analizar o una instancia de `java.io.Reader`. Este puede ser especificado directamente en el cuerpo de la etiqueta `<c:parse>` sin usar el atributo correspondiente.

var: Nombre de la variable que referencia al documento XML.

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

filter: Se especifica una instancia de la clase `org.xml.sax.XMLFilter` para filtrar el documento antes de analizarlo. Este atributo es especialmente útil si el documento que se va a analizar es muy grande y solo se requiere una pequeña porción de este.

systemId: Indica la URI para el documento que se analiza y resuelve cualquier ruta relativa que esté presente en el mismo. Este atributo es necesario si el código XML que se está analizando utiliza direcciones URL relativas para hacer referencia a otros documentos o recursos a los cuales se necesita tener acceso durante el proceso de análisis.

Si se requiere realizar operaciones sobre el documento analizado, el cual debe adherirse a un interfaz estándar, la sintaxis de la etiqueta es la siguiente:

Sintaxis 1

```
<x:parse xml="xml" varDom="name" scopeDom="scope" filter="expression" systemId="expression"/>
```


Sintaxis 2

```
<x:parse varDom="name" scopeDom="scope" filter="expression" systemId="expression">  
  xml  
</x:parse>
```

Cuando se utiliza esta versión de `<x:parse>`, el objeto que representa el documento XML analizado debe implementar la interfaz `org.w3c.dom.Document`.

varDom: Nombre de la variable que referencia al documento XML.

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

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

Trabajando con el contenido XML

A menudo sólo ciertos elementos de los datos contenidos en el documento XML serán de interés para una aplicación particular. Por este motivo, la librería incluye varias etiquetas XML para acceder y manipular los elementos individuales de los documentos XML.

Estas etiquetas se basan en las etiquetas de la librería "Core", si bien estas últimas utilizan expresiones EL, sus homólogos para manipular los documentos XML usan expresiones XPath.

XPath es una notación estándar para hacer referencia a los elementos de un documento XML, sus atributos y contenido. Como su nombre lo indica, esta notación se asemeja a las rutas de archivos del sistema en el sentido de que los componentes de una declaración de XPath se delimitan mediante barras. Estos componentes se mapean con nodos del documento XML. Además, los asteriscos se pueden utilizar como comodines para considerar varios nodos, y las expresiones entre corchetes se pueden usar para que coincidan con los valores de los atributos, en este último caso es requerido especificar índices.

Para mostrar los datos de un elemento del documento XML, se utiliza la etiqueta `<x:out>`, que es análoga a `<c:out>`. Los atributos de esta etiqueta son "select" y "escapeXml".

`<x:out select="XPathExpression" escapeXml="boolean"/>`

La diferencia, por supuesto, es que el valor del atributo "select" debe ser una expresión XPath, mientras que el atributo "value" de `<c:out>` debe ser una expresión EL. El significado del atributo escapeXml es el mismo en ambas etiquetas.

Además de `<x:out>`, la librería incluye las siguientes etiquetas para la manipulación de datos XML:

- `<x:set>` para asignar el valor de una expresión XPath a una variable JSTL bajo un determinado alcance.
- `<x:if>` para evaluar una expresión XPath.
- `<x:choose>`, `<x:when>` y `<x:otherwise>` para evaluar distintas expresiones XPath mutuamente excluyentes.
- `<x:forEach>` para iterar sobre los múltiples elementos de una expresión XPath.

En todos los casos, se utiliza el atributo "select" en el cual se debe especificar una expresión XPath.

Veamos algunos ejemplos,

Ejemplo 1

Página JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplo 1</title>
    <link rel="stylesheet" type="text/css" href="./css/style.css"/>
</head>
<body>
    <c:catch var="exception">
        <c:import var="rssFeed" url="https://news.google.com.ar/?output=rss"/>
        <x:parse var="r" xml="${rssFeed}"/>
        <h1>
            <x:out select="$r/rss/channel/title" />
        </h1>
        <x:forEach select="$r/rss/channel/item">
            <x:out select="description" escapeXml="false"/>
        </x:forEach>
    </c:catch>
    <c:if test="${!empty exception}">
        <p>${exception}</p>
    </c:if>
    <br/>
    <a href="index.jsp" target="_self">Volver al index principal</a>
</body>
</html>
```

Ejemplo 2

Archivo XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cereales xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./cereales.xsd">
  <cereal>Cebada</cereal>
  <cereal>Avena</cereal>
  <cereal>Mijo</cereal>
  <cereal>Maíz</cereal>
  <cereal>Arroz</cereal>
  <cereal>Centeno</cereal>
  <cereal>Trigo</cereal>
</cereales>
```

Página JSP

```
<%@ page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplo</title>
  </head>
  <body>
    <h1>Cereales</h1>
    <c:import var="cereales" url="/WEB-INF/cereales.xml"/>
    <x:parse var="cer" xml="{cereales}"/>
    <ol>
      <x:forEach var="c" select="$cer/cereales/cereal">
        <li><x:out select="$c" escapeXml="false" /></li>
      </x:forEach>
    </ol>
  </body>
</html>
```

Ejemplo 3

Archivo XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<productos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./stock.xsd">
  <producto>
    <id>1</id>
    <nom>Lapicera</nom>
    <cant>10</cant>
  </producto>
  <producto>
    <id>2</id>
    <nom>Plasticola</nom>
    <cant>4</cant>
  </producto>
  <producto>
    <id>3</id>
    <nom>Lápiz</nom>
    <cant>0</cant>
  </producto>
  <producto>
    <id>4</id>
    <nom>Resma de Papel A4</nom>
    <cant>0</cant>
  </producto>
```

</productos>

Página JSP

```
<%@ page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplo</title>
  </head>
  <body>
    <h1>Productos faltantes</h1>
    <c:import var="stock" url="/WEB-INF/stock.xml"/>
    <x:parse var="st" xml="{stock}"/>
    <table>
      <tr>
        <th>Id.</th>
        <th>Nombre</th>
      </tr>
      <x:forEach var="s" select="$st/productos/producto">
        <x:if select="$s/cant = 0">
          <tr>
            <td><x:out select="$s/id" escapeXml="false" /></td>
            <td><x:out select="$s/nom" escapeXml="false" /></td>
          </tr>
        </x:if>
      </x:forEach>
    </table>
  </body>
</html>
```

Transformando XML

Para poder transformar XML se utilizan hojas de estilo XSL. JSTL apoya esta operación mediante el uso de la etiqueta <x:transform>. Como fue el caso de <x:parse>, la etiqueta <x:transform> admite varias formas diferentes. Entre ellas:

Sintaxis 1

```
<x:transform xml="xml" xslt="expression" var="name" scope="scope" xmlSystemId="expression"
xsltSystemId="expression">
  <x:param name="expression" value="expression"/>
  ...
</x:transform>
```

Sintaxis 2

```
<x:transform xslt="expression" var="name" scope="scope" xmlSystemId="expression"
xsltSystemId="expression">
  xml
  <x:param name="expression" value="expression"/>
  ...
</x:transform>
```

xml: Su valor debe ser una cadena que contiene el documento XML a transformar o una instancia de java.io.Reader. También puede ser una instancia de la clase org.w3c.dom.Document o javax.xml.transform.Source., o finalmente, puede ser el valor de una variable asignada a través del atributo "var" o "varDom" de la etiqueta <x:parse>. A su vez, el valor puede ser especificado directamente en el cuerpo de la etiqueta <c:transform> sin usar el atributo correspondiente.

xslt: Define la hoja de estilo con la que se transforma el documento XML. Este atributo es obligatorio.

var: Nombre de la variable que referencia al documento XML ya transformado.

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

xml:SystemId: Indica la URI para el documento que se transforma y resuelve cualquier ruta relativa que esté presente en el mismo. Este atributo es necesario si el código XML que se está transformando utiliza direcciones URL relativas para hacer referencia a otros documentos o recursos a los cuales se necesita tener acceso durante el proceso de transformación.

xslt:SystemId: Indica la URI para la hoja de estilos y resuelve cualquier ruta relativa que esté presente en la misma. Este atributo es necesario si el código XSL presenta direcciones URL relativas que hacen referencia a otras hojas de estilos a las cuales se necesita tener acceso.

Si el manejador de hojas de estilos para la transformación del documento recibe parámetros, estos deben ser especificados a través de la etiqueta `<x:param>`.

Un detalle a tener en cuenta, es que si se especifica el documento XML en el cuerpo de la etiqueta `<x:transform>`, este debe aparecer antes que la definición de parámetros.

La etiqueta `<x:param>` posee dos atributos "name" y "value", ambos obligatorios.

De los ejemplos antes desarrollados,

Ejemplo 1

Archivo XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo 3</title>
        <link rel="stylesheet" type="text/css" href="./css/style.css"/>
      </head>
      <body>
        <h1>
          <xsl:value-of select="/rss/channel/title" />
        </h1>
        <ul>
          <xsl:for-each select="/rss/channel/item">
            <li>
              <a target="_blank">
                <xsl:attribute name="href">
                  <xsl:value-of select="./link"/>
                </xsl:attribute>
                <xsl:value-of select="./title" />
              </a>
            </li>
          </xsl:for-each>
        </ul>
        <br/>
        <a href="index.jsp" target="_self">Volver al index principal</a>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Página JSP

```
<%@ page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
<!DOCTYPE html>
<c:catch var="exception">
  <c:import var="rssFeed" url="https://news.google.com.ar/?output=rss"/>
  <c:import var="rssFeedHtml" url="/WEB-INF/rss.xml"/>
  <x:transform xml="${rssFeed}" xslt="${rssFeedHtml}"/>
</c:catch>
<c:if test="${!empty exception}">
  <p>${exception}</p>
</c:if>
```

Ejemplo 2

Archivo XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo 4</title>
        <link rel="stylesheet" type="text/css" href="./css/style.css"/>
      </head>
      <body>
        <h1>Productos faltantes</h1>
        <table class="tabla">
          <thead>
            <tr>
              <td align="right">Id.</td>
              <td>Nombre</td>
            </tr>
          </thead>
          <tbody>
            <xsl:for-each select="/productos/producto">
              <xsl:if test="./cant = 0">
                <tr>
                  <td align="right"><xsl:value-of select="./id"/></td>
                  <td><xsl:value-of select="./nom"/></td>
                </tr>
              </xsl:if>
            </xsl:for-each>
          </tbody>
        </table>
        <br/>
        <a href="index.jsp" target="_self">Volver al index principal</a>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Página JSP

```
<%@ page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
<!DOCTYPE html>
<c:catch var="exception">
  <c:import var="stock" url="/WEB-INF/stock.xml"/>
  <c:import var="stockHtml" url="/WEB-INF/stock.xsl"/>
  <x:transform xml="${stock}" xslt="${stockHtml}"/>
</c:catch>
<c:if test="${!empty exception}">
  <p>${exception}</p>
</c:if>
```