JSTL (JSP Standard Tag Library)

JSP - Expression Language (EL)

Introduction

•Expression Language was first introduced in JSTL 1.0 (JSP Standard Tag Library).

•Before the introduction of JSTL, scriptlets were used to manipulate application data.

- JSTL introduced the concept of an expression language (EL) which simplified the page development by providing standard tag libraries.
- These tag libraries provide support for common, structural tasks, such as: iteration and conditionals, processing XML documents, internationalization and database access using the Structured Query Language (SQL).

The Expression Language introduced in JSTL 1.0 is now incorporated in JavaServer Pages specification(JSP 2.0).

This presentation gives some idea about what is Expression Language and how to simplify the maintenance for JSP applications by avoiding scripting elements.

JSP - Expression Language (EL)

- A primary feature of JSP technology version 2.0 is its support for an expression language (EL).
- An expression language makes it possible to easily access application data stored in JavaBeans components.

For example, the JSP expression language allows a page author to access a bean using simple syntax such as

${name} for a simple variable

or

${name.foo.bar} for a nested property.

- JSP EL allows you to create expressions both **(a)** arithmetic and **(b)** logical. Within a JSP EL expression, you can use integers, floating point numbers, strings, the built-in constants true and false for boolean values, and null.

JSP Standard Tag Library

- The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.
- JSTL allows you to program your JSP pages using tags, rather than the scriptlet code that most JSP programmers are already accustomed to. JSTL can do nearly everything that regular JSP scriptlet code can do.

Advantage of JSTL

- **Fast Development** JSTL provides many tags that simplify the JSP.
- **Code Reusability** We can use the JSTL tags on various pages.
- **No need to use scriptlet tag** It avoids the use of scriptlet tag.

```
<html>
<head>
<title>Count to 10 in JSP scriptlet</title>
</head>
<body>
<% for(int i=1;i<=10;i++)
    {%>
   <%=i%><br/>
<%
    }
%>
</body>
</html>
```

```
<%@ taglib uri="http://java.sun.com/jstl/core"
prefix="c" %>
<html>
<head>
<title>Count to 10 Example (using JSTL)</title>
</head>

<body>
<c:forEach var="i" begin="1" end="10"
step="1">
<c:out value="${i}" />

<br />
</c:forEach>
</body>
</html>
```

THE JSTL TAG LIBRARIES

The JSTL tags can be classified, according to their functions, into following JSTL tag library groups that can be used when creating a JSP page:

- **Core Tags**
- **Formatting tags**
- **SQL tags**
- **XML tags**
- **JSTL Functions**

THE JSTL TAG LIBRARIES

- **Core Tag Library**—Contains tags that are essential to nearly any Web application. Examples of core tag libraries include looping, expression evaluation, and basic input and output.

- **Formatting/Internationalization Tag Library**—Contains tags that are used to parse data. Some of these tags will parse data, such as dates, differently based on the current locale.

- **Database Tag Library**—Contains tags that can be used to access SQL databases. These tags are normally used only to create prototype programs. This is because most programs will not handle database access directly from JSP pages. Database access should be embedded in EJBs that are accessed by the JSP pages.

- **XML Tag Library**—Contains tags that can be used to access XML elements. Because XML is used in many Web applications, XML processing is an important feature of JSTL.

- **JSTL Functions --**JSTL includes a number of standard functions, most of which are common string manipulation functions.

Core Tags:

The core group of tags are the most frequently used JSTL tags. Following is the syntax to include JSTL Core library in your JSP:

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

T

Tag	Description
<c:out >	Like <%= ... >, but for expressions.
<c:set >	Sets the result of an expression evaluation in a 'scope'
<c:remove >	Removes a scoped variable (from a particular scope, if specified).
<c:catch>	Catches any Throwable that occurs in its body and optionally exposes it.
<c:if>	Simple conditional tag which evalutes its body if the supplied condition is true.
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<c:when>	Subtag of <choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'.
<c:import>	Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'.
<c:forEach >	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality .
<c:forTokens>	Iterates over tokens, separated by the supplied delimeters.
<c:param>	Adds a parameter to a containing 'import' tag's URL.
<c:redirect >	Redirects to a new URL.
<c:url>	Creates a URL with optional query parameters

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

JSTL Functions:

JSTL includes a number of standard functions, most of which are common string manipulation functions. Following is the syntax to include JSTL Functions library in your JSP:

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

Following is the list of JSTL Functions:

Function	Description
fn:contains()	Tests if an input string contains the specified substring.
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way.
fn:endsWith()	Tests if an input string ends with the specified suffix.
fn:escapeXml()	Escapes characters that could be interpreted as XML markup.
fn:indexOf()	Returns the index within a string of the first occurrence of a specified substring.
fn:join()	Joins all elements of an array into a string.
fn:length()	Returns the number of items in a collection, or the number of characters in a string.
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string.
fn:split()	Splits a string into an array of substrings.

Formatting tags:

The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Web sites. Following is the syntax to include Formatting library in your JSP:

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

Following is the list of Formatting JSTL Tags:

Tag	Description
<fmt:formatNumber>	To render numerical value with specific precision or format.
<fmt:parseNumber>	Parses the string representation of a number, currency, or percentage.
<fmt:formatDate>	Formats a date and/or time using the supplied styles and pattern
<fmt:parseDate>	Parses the string representation of a date and/or time
<fmt:bundle>	Loads a resource bundle to be used by its tag body.
<fmt:setLocale>	Stores the given locale in the locale configuration variable.
<fmt:setBundle>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable.
<fmt:timeZone>	Specifies the time zone for any time formatting or parsing actions nested in its body.
<fmt:setTimeZone>	Stores the given time zone in the time zone configuration variable
<fmt:message>	To display an internationalized message.
<fmt:requestEncoding>	Sets the request character encoding

SQL tags:

- The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as Oracle, mySQL, or Microsoft SQL Server.
- Following is the syntax to include JSTL SQL library in your JSP:

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

Following is the list of SQL JSTL Tags:

Tag	Description
<sql:setDataSource>	Creates a simple DataSource suitable only for prototyping
<sql:query>	Executes the SQL query defined in its body or through the sql attribute.
<sql:update>	Executes the SQL update defined in its body or through the sql attribute.
<sql:param>	Sets a parameter in an SQL statement to the specified value.
<sql:dateParam>	Sets a parameter in an SQL statement to the specified java.util.Date value.
<sql:transaction >	Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.

sql:update

```
<sql:update dataSource="${db}" var="count">
INSERT INTO Students VALUES (154,'Nasreen'
   , 'jaha', 25);
</sql:update>
```

sql:param

```
<c:set var="StudentId" value="152"/>
<sql:update dataSource="${db}" var="count">
DELETE FROM Students WHERE Id = ?
<sql:param value="${StudentId}" />
</sql:update>
```

sql:dateParam

```
<%
Date DoB = new Date("2000/10/16");
int studentId = 151;
%>
<sql:update dataSource="${db}" var="count">
  UPDATE Student SET dob = ? WHERE Id = ?

  <sql:dateParam value="
  <%=DoB%>" type="DATE" />
  <sql:param value="<%=studentId%>" />
```

sql:transaction

```
<sql:transaction dataSource="${db}">
  <sql:update var="count">
    UPDATE Student SET First_Name = 'Suraj'
   WHERE Id = 150
  </sql:update>
  <sql:update var="count">
    UPDATE Student SET Last_Name= 'Saifi' W
   HERE Id = 153
  </sql:update>
  <sql:update var="count">
```

XML TAGS:

- The JSTL XML tags provide a JSP-centric way of creating and manipulating XML documents. Following is the syntax to include JSTL XML library in your JSP:

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

- The JSTL XML tag library has custom tags for interacting with XML data. This includes parsing XML, transforming XML data, and flow control based on XPath expressions.

Following is the list of XML JSTL Tags:

Tag	Description
<x:out>	Like <%= ... >, but for XPath expressions.
<x:parse>	Use to parse XML data specified either via an attribute or in the tag body.
<x:set >	Sets a variable to the value of an XPath expression.
<x:if >	Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored.
<x:forEach>	To loop over nodes in an XML document.
<x:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<x:when >	Subtag of <choose> that includes its body if its expression evalutes to 'true'
<x:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'
<x:transform >	Applies an XSL transformation on a XML document
<x:param >	Use along with the transform tag to set a parameter in the XSLT stylesheet

x:set

```
<c:set var="book">
<books>
<book>
  <name>Three mistakes of my life</name>
  <author>Chetan Bhagat</author>
  <price>200</price>
</book>
<book>
  <name>Tomorrow land</name>
  <author>Brad Bird</author>
  <price>2000</price>
</book>
</books>
</c:set>

<x:set var="fragment" select="$output/books/book[2]/price"/>
<b>The price of the Tomorrow land book</b>:
<x:out select="$fragment" />
```

<x:choose>, <x:when>, <x:otherwise>

```
<c:set var="xmltext">
<books>
<book>
  <name>Three mistakes o
    f my life</name>
  <author>Chetan Bhagat
    </author>
  <price>200</price>
</book>
<book>
  <name>Tomorrow land<
```

```
<x:parse xml="${xmltext
    }" var="output"/>
<x:choose>
  <x:when select="$outpu
   t//book/author = 'Chetan
    bhagat'">
     Book is written by Che
   tan bhagat
  </x:when>
  <x:when select="$outpu
   t//book/author = 'Brad B
```

x:if

```
<x:if select="$output/vegetables/vegetable/price
   < 100">
   Vegetables prices are very low.
</x:if>
```

x:transform

<x:transform xml="${xml}" xslt="${xsl}" />

x:param

```
<c:import url="transfer.xsl" var="xslt"/>
<x:transform xml="${xmltext}" xslt="${xslt}"
  >
  <x:param name="bgColor" value="yellow"/>

</x:transform>
```