JSP IV

In this topic we will discuss about, Expression Language(EL) - need, advatages and disadvantages, JSP Standard Tag Library(JSTL) - need and advantages and custom tags - a basic example.

EL

Expression language (EL) has been introduced in JSP 2.0. The main purpose of it to simplify the process of accessing data from bean properties and from implicit objects. EL includes arithmetic, relational and logical operators too.

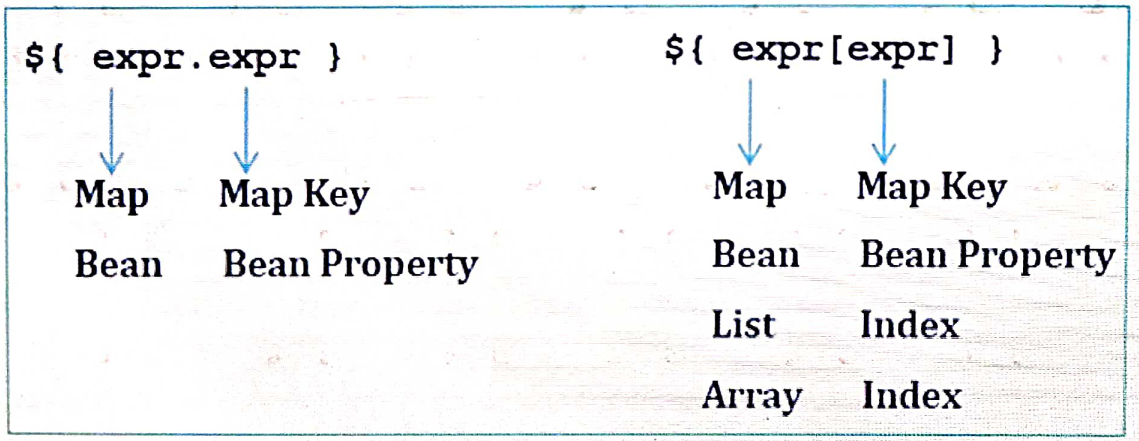
Synatx of EL:

1. ${expression}
2. ${ expr.expr }
3. ${ expr[expr] }

Whatever present inside braces gets evaluated at runtime and being sent to the output stream.

In EL there are two access operators.

1. . (dot) operator
2. [ ] (bracket) operator



Advantages of EL

- EL has more elegant and compact syntax than standard JSP tags
- EL lets you access nested properties

Advantages of EL

- EL has more elegant and compact syntax than standard JSP tags
- EL lets you access nested properties
- EL let you access collections such as maps, arrays, and lists
- EL does a better job of handling null values
- EL does not make it compulsory to specify the scope of the bean. If scope is not mentioned, it will search the bean from the least accessible scope to the most accessible scope.
- EL provides it's own set of implicit objects which allows to access data from request parametera, request headers, cookies, context parameters etc

Disadvantages of EL

- EL doesn't create a JavaBean if it doesn't exist
- EL doesn't provide a way to set properties.
- EL do not provide iteration over the bean if it is type of collection

Different ways to use EL

1. EL evaluates the simple expressions

```
<body>
${1<2}
${1+2+3}
${"welcome"}
</body>
```

Output:

true 6 welcome

2. EL access an attribute name

Servlet code

```
Date currentDate = new Date();
request.setAttribute("currentDate", currentDate)
```

JSP code

```
<p> The current date is ${currentDate}</p>
```

2. EL access an attribute name

Servlet code

```
Date currentDate = new Date();
request.setAttribute("currentDate", currentDate)
```

JSP code

```
<p> The current date is ${currentDate}</p>
```

3. EL access the property and even a nested property of an attribute

- Servlet code

```
User user = new User(firstName, lastName, emailAddress, address);
session.setAttribute("user", user)
```

- JSP code

```
<p> Hello ${user.firstName}</p>
<p> City : ${user.address.city} </p>
```

4. EL can use implicit objects for fetching information from other sources

The list of implicit objects provided by EL is as follows :

EL implicit object	Usage
param	It maps the request parameter to the single value
paramValues	It maps the request parameter to an array of values
header	It maps the request header name to the single value
headerValues	It maps the request header name to an array of values
cookie	It maps the given cookie name to the cookie value
initParam	It maps the initialization parameter
pageScope	It maps the given attribute name with the value set in the page scope
requestScope	It maps the given attribute name with the value set in the request scope
sessionScope	It maps the given attribute name with the value set in the session scope
applicationScope	It maps the given attribute name with the value set in the application scope
pageContext	It provides access to many objects request, session etc.

4. EL can use implicit objects for fetching information from other sources

The list of implicit objects provided by EL is as follows :

EL implicit object	Usage
param	It maps the request parameter to the single value
paramValues	It maps the request parameter to an array of values
header	It maps the request header name to the single value
headerValues	It maps the request header name to an array of values
cookie	It maps the given cookie name to the cookie value
initParam	It maps the initialization parameter
pageScope	It maps the given attribute name with the value set in the page scope
requestScope	It maps the given attribute name with the value set in the request scope
sessionScope	It maps the given attribute name with the value set in the session scope
applicationScope	It maps the given attribute name with the value set in the application scope
pageContext	It provides access to many objects request, session etc

Examples :

EL	Meaning
Welcome, ${ param.name }	Here name is the name of the request parameter
Value is ${ sessionScope.user }	Here user is the name of the bean set at session scope
Hello, ${cookie.name.value}	Here name is the name of the cookie
Please contact ${initParam.adminemail}	Here adminemail is the name of the context parameter set in web.xml
${header["host"]}	Here host is the name of the request header
${pageContext.session.id}	For accessing session, pageContext implicit object is needed

Use [ ] operator to work with arrays and lists

${attribute[“propertyKeyOrIndex”]}

Servlet code

```
String[ ] colors = {“Red”, “Green”, “Blue”};
ServletContext application = this.getServletContext();
application.setAttribute(“colors”, colors);
```

Use [ ] operator to work with arrays and lists

${attribute["propertyKeyOrIndex"]}

Servlet code

```
String[ ] colors = {"Red", "Green", "Blue"};
ServletContext application = this.getServletContext();
application.setAttribute("colors", colors);
```

JSP code

```
<p> The first color is ${colors[0]}<br>
The second color is ${colors[1]}</p>
```

Another way to write JSP code

```
<p> The first color is ${colors["0"]}<br>
The second color is ${colors["1"]}</p>
```

EL Operators

- Arithmetic : + - * /(div) % (mod)
- Relational : == (eq) != (ne)< (lt) > (gt) <=(le) >=(ge)
- Logical : &&(and) || (or) ! (Not)

But use extremely sparingly to preserve MVC model

Preventing EL Evaluation

If JSP page contains ${ then it is required to deactivate EL from JSP page

```
<%@ page isELIgnored="true" %>
```

Above page directive can be used to deactivate EL in individual JSP page

```
<jsp-config>
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<el-ignored>true</el-ignored>
</jsp-property-group
</jsp-config>
```

Preventing EL Evaluation

If JSP page contains ${ then it is required to deactivate EL from JSP page

```
<%@ page isELIgnored="true" %>
```

Above page directive can be used to deactivate EL in individual JSP page

```
<jsp-config>
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<el-ignored>true</el-ignored>
</jsp-property-group
</jsp-config>
```

Above configuration in web.xml deactivates the EL in an entire Web application.

EL has got many advantages over std actionsbut its fails iterate over the bean of type collection

---

JSTL

JSP Standard Tag Library(JSTL) is a standard library of readymade tags. The JSTL contains several tags that can remove scriplet code from a JSP page by providing some ready to use, already implemented common functionalities.

To use JSTL in our JSP pages, we need to download the JSTL jars for your servlet container. Most of the times, you can find them in the example projects of server download and you can use them. You need to include these libraries in your web application project WEB-INF/lib directory.

JSTL is divided into 5 groups:

1. JSTL Core: JSTL Core provides several core tags such as if, forEach, import, out etc to support some basic scripting task. Url to include JSTL Core Tag inside JSP page is :
   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2. JSTL Formatting: JSTL Formatting library provides tags to format text, date, number for Internationalised web sites. Url to include JSTL Formatting Tags inside JSP page is :
   <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
3. JSTL sql: JSTL SQL library provides support for Relational Database Connection and tags to perform operations like insert, delete, update, select etc on SQL databases. Url to include JSTL SQL Tag inside JSP page is :
   <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

JSTL is divided into 5 groups:

1. JSTL Core: JSTL Core provides several core tags such as if, forEach, import, out etc to support some basic scripting task. Url to include JSTL Core Tag inside JSP page is :
   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2. JSTL Formatting: JSTL Formatting library provides tags to format text, date, number for Internationalised web sites. Url to include JSTL Formatting Tags inside JSP page is :
   <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
3. JSTL sql: JSTL SQL library provides support for Relational Database Connection and tags to perform operations like insert, delete, update, select etc on SQL databases. Url to include JSTL SQL Tag inside JSP page is :
   <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
4. JSTL XML: JSTL XML library provides support for XML processing. It provides flow control, transformation features etc. Url to include JSTL XML Tag inside JSP page is :
   <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
5. JSTL functions: JSTL functions library provides support for string manipulation. Url to include JSTL Function Tag inside JSP page is :
   <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

JSTL core tags

JSTL core tag	Description
<c:import>	Same as jsp:include or include directive
<c:redirect>	redirect request to another resource
<c:set>	To set the variable value in given scope.
<c:remove>	To remove the variable from given scope
<c:catch>	To catch the exception and wrap it into an object.
<c:if>	Simple conditional logic, used with EL and we can use it to process the exception from <c:catch>
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <c:when> and <c:otherwise>
<c:when>	Subtag of <c:choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise>	Subtag of <c:choose> that includes its body if its condition evalutes to 'false'.
<c:forEach>	for iteration over a collection

JSTL core tags

JSTL core tag	Description
<c:import>	Same as jsp:include or include directive
<c:redirect>	redirect request to another resource
<c:set>	To set the variable value in given scope.
<c:remove>	To remove the variable from given scope
<c:catch>	To catch the exception and wrap it into an object.
<c:if>	Simple conditional logic, used with EL and we can use it to process the exception from <c:catch>
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <c:when> and <c:otherwise>
<c:when>	Subtag of <c:choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise>	Subtag of <c:choose> that includes its body if its condition evalutes to 'false'.
<c:forEach>	for iteration over a collection

Examples :

1. Using conditional core tag

```
<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>
```

Output :

My income is: 16000

2. Using iteration tag

```
<body>
<table border="1">
<c:forEach var="emp" items="${sessionScope.employees}">
  <tr>
    <td>${emp.empid}</td>
```

Examples :

1. Using conditional core tag

```
<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>
```

Output :

My income is: 16000

2. Using iteration tag

```
<body>
<table border="1">
<c:forEach var="emp" items="${sessionScope.employees}">
  <tr>
    <td>${emp.empid}</td>
    <td>${emp.ename}</td>
    <td>${emp.salary}</td>
  </tr>
</c:forEach>
</table>
</body>
```

Output :

1001	Rahul	10000
1004	Vishal	12000

Follow the steps for using JSTL in JSP page

1. Add the JSTL jar files. Jar files should be available in WEB-INF/lib folder after deployment of the application
2. Add taglib directive to specify URI of the tag library and prefix

Follow the steps for using JSTL in JSP page

1. Add the JSTL jar files. Jar files should be available in WEB-INF/lib folder after deployment of the application
2. Add taglib directive to specify URI of the tag library and prefix
3. Use required tags from JSTL subgroups with the specified prefix

---

Custom Tags

A custom tag library is a set of custom tags that invoke custom actions in a JavaServer Pages (JSP) file. Tag libraries move the functionality provided by the tags into tag implementation classes, reducing the task of embedding excessive amounts of Java™ code in JSP pages.

Benefits of tag libraries

1. Help separate presentation from implementation.
2. Are easy to maintain and reuse
3. Simplify complex actions
4. Provide Java coded functions without the task of coding in Java.
5. Can dynamically generate page content and implement a controlled flow.

Syntax

The custom tag can be used in two ways. They are as follows:

1. <prefix:tag name attr1=value1.... attr = value />
2. <prefix:tag name attr1=value1.... attr = value>
   body code
   </prefix:tagname>

Steps for implementing custom tags

1. Tag handler class: We define what our custom tag will do when used in a JSP page in this class.
   This is the first step in using JSP to create custom tags. We inherit the TagSupport class and override the function doStartTag() to create the Tag Handler. The JspWriter class is used to write data to the JSP.
   The getOut() method of the PageContext class returns an instance of the JspWriter class. By default, the TagSupport class gives a pageContext instance.

   package com.customtags;

Steps for implementing custom tags

1. Tag handler class: We define what our custom tag will do when used in a JSP page in this class.
   This is the first step in using JSP to create custom tags. We inherit the TagSupport class and override the function doStartTag() to create the Tag Handler. The JspWriter class is used to write data to the JSP.
   The getOut() method of the PageContext class returns an instance of the JspWriter class. By default, the TagSupport class gives a pageContext instance.

```
package com.customtags;
import java.util.Calendar;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;
public class MyTagHandler extends TagSupport{
public int doStartTag() throws JspException {
  JspWriter out=pageContext.getOut();//JspWriter's instance is returned.
  try{
      out.print(Calendar.getInstance().getTime());//JspWriter is used to print the date and time.
  }catch(Exception e){System.out.println(e);}
      return SKIP_BODY;//The tag's body content will not be evaluated.
  }
}
```

2. Tag descriptor file (TLD): It contains the tag name, tag handler class, and tag attributes.
   A TLD file should be created in the WEB-INF directory, which will be loaded by the container when the application is deployed. Tag and Tag Handler classes are described in the Tag Library Descriptor (TLD) file.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
     PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
  "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
 <tlib-version>1.0</tlib-version>
 <jsp-version>1.2</jsp-version>
 <short-name>simple</short-name>
 <uri>http://tomcat.apache.org/example-taglib</uri>
 <tag>
```

2. Tag descriptor file (TLD): It contains the tag name, tag handler class, and tag attributes.
   A TLD file should be created in the WEB-INF directory, which will be loaded by the container when the application is deployed. Tag and Tag Handler classes are described in the Tag Library Descriptor (TLD) file.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
      PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
   "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
 <tlib-version>1.0</tlib-version>
 <jsp-version>1.2</jsp-version>
 <short-name>simple</short-name>
 <uri>http://tomcat.apache.org/example-taglib</uri>
 <tag>
  <name>today</name>
  <tag-class>com.customtags.MyTagHandler</tag-class>
  <!-- <attribute>
  <name> </name>
    <required> </required>
    <rtexprevalue> </rtexprvalue>
  </attribute>
 </tag>  -->
</taglib>
```

   Please note that attribute tag cab be repeated for as many number of attributes required for tag.

3. JSP page: This is the JSP page where we'll use our custom tag.
   For using custom tags, we need to add taglib directive on JSP page. URI attribute of taglib directive can have path of tld path instead of URI of tag library

```
<%@ taglib uri="WEB-INF/mytags.tld" prefix="m" %>
Current Date and Time is: <m:today/>
```

Assignments

1. Practice EL for displaying data from attributes. Use EL implicit objects for displaying data from request parameters, request headers, cookies and context paramteres.
2. Extend blogging application with following features

```
<name>today</name>
<tag-class>com.customtags.MyTagHandler</tag-class>
<!-- <attribute>
<name> </name>
  <required> </required>
  <rtexprevalue> </rtexprvalue>
</attribute>
 </tag>  -->
</taglib>
```

Please note that attribute tag cab be repeated for as many number of attributes required for tag.

3. JSP page: This is the JSP page where we'll use our custom tag.
   For using custom tags, we need to add taglib directive on JSP page. URI attribute of taglib directive can have path of tld path instead of URI of tag library

```
<%@ taglib uri="WEB-INF/mytags.tld" prefix="m" %>
Current Date and Time is: <m:today/>
```

Assignments

1. Practice EL for displaying data from attributes. Use EL implicit objects for displaying data from request parameters, request headers, cookies and context paramteres.
2. Extend blogging application with following features
   a. Display all the available topics to the user
   b. Display all the comments from all uers on the selected topic.
   c. Allow the user to add the the topic - this should be allowed only after login
   d. Allow the user to add the comment on the selected topic- this should be allowed only after login
   Please Note :
   a. Use MVC architecture for request processing
   b. Establish the connection in listener
   c. Write no java in JSP and no HTML in servlet
3. Design a greet tag as custom tag which will be used to give "welcome <name>" message to the user with the message like 'Good Morning' based on time