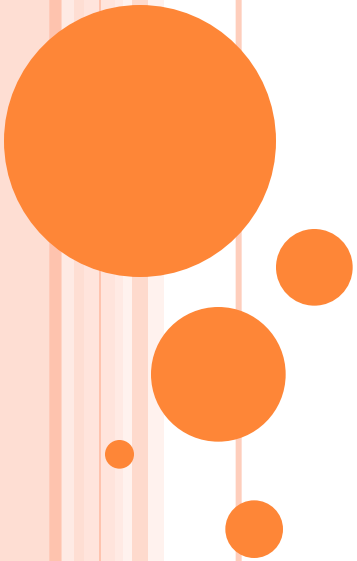

JSTL

JSP Standard Tag Library

7-Jan-21

What is JSTL?

- **JSTL (JSP Standard Tag Libraries)** is a collection of JSP custom tags developed by Java Community Process, www.jcp.org. The reference implementation is developed by the Jakarta project, jakarta.apache.org.
- **Full JSTL**
  - Contains many common and useful JSP custom tags
  - Particularly useful when you are using MVC, but the data contains a varying number of entries
  - Based on the Struts looping and logic tags.
- 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.

COUNT TO TEN EXAMPLE USING SCRIPTLET:

- JSTL was introduced to allow JSP programmers to program using tags rather than Java code.
- To show why this is preferable, a quick example is in order. We will examine a very simple JSP page that counts to ten. **(count1.jsp)**
- We will examine this page both as regular scriptlet-based JSP, and then as JSTL. When the count to ten example is programmed using scriptlet based JSP, the JSP page appears as follows.`

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

- As you can see from the preceding example, using scriptlet code produces page source code that contains a mix of HTML tags and Java statements.

- The primary reason that it is not optimal to mix scriptlet and tag-based code is readability. JSTL allows the human programmer to look at a program that consists entirely of HTML and HTML-like tags.

COUNT TO TEN EXAMPLE USING JSTL:

The following code shows how the count to ten example would be written using JSTL. As you can see, this code listing is much more constant, as only tags are used. HTML and JSTL tags are mixed to produce the example.

count2.jsp

```
<%@ 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>
```

Count3.jsp – JSTL and GET

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

</HEAD>
<BODY>
  <c:set var="n" value="${param.number}" />
  Hello ...Printing numbers till N=
  <c:out value="${n}"/>
  <br/>
  <br/>

  <c:forEach var="i" begin="1" end="${n}" step="1">
       <c:out value="${i}"/>
  </c:forEach>
</BODY>
```

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 (c:xxx)**
- Contains tags that are essential to nearly any Web application. Examples of core tag libraries include looping, expression evaluation, and basic input and output.

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" %>

There are following Core JSTL Tags:

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

JSP - Expression Language (EL)

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

  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 from objects/bean
- 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.

The test attribute of the following conditional tag is supplied with an EL expression that compares the number of items in the session-scoped bean named cart with 0:

```
<c:if test="${sessionScope.cart.numberOfItems > 0}">   ... </c:if>
```

The JSP expression evaluator is responsible for handling EL expressions, which are enclosed by the ${ } characters and can include literals. Here's an example:

```
<c:if test="${bean1.a < 3}" >   ... </c:if>
```

Any value that does not begin with ${ is treated as a literal and is parsed to the expected type using the PropertyEditor for the type:

```
<c:if test="true" > ... </c:if>
```

Literal values that contain the ${ characters must be escaped as follows:

```
<mytags:example attr1="an expression is ${'${'}true}" />
```

Basic Operators in EL:

JSP Expression Language (EL) supports most of the arithmetic and logical operators supported by Java. Below is the list of most frequently used operators:

Operator	Description
.	Access a bean property or Map entry
[]	Access an array or List element
( )	Group a subexpression to change the evaluation order
+	Addition
-	Subtraction or negation of a value
*	Multiplication
/ or div	Division
% or mod	Modulo (remainder)
== or eq	Test for equality
!= or ne	Test for inequality
< or lt	Test for less than
> or gt	Test for greater than
<= or le	Test for less than or equal
>= or gt	Test for greater than or equal
&& or and	Test for logical AND
|| or or	Test for logical OR
! or not	Unary Boolean complement
empty	Test for null, empty String, array or Collection.
func(args)	A function call

Arithmetic operators

Before JSP2.0/JSTL

```
<%!
        int k = 0;
%>
<%
        k = 10 + 20;
        out.println("value of k is.."+k);
%>
```

Using JSP2.0/JSTL (without using scriptlet)

```
value of k is ${10 + 20}
```

Few more examples on arithmetic operations.

EL Expression	Result
${10.2 + 20.3}	O/P: 30.5
${-40 - 20}	O/P: -60
${20 * 2}	O/P: 40
${3/4}	O/P: 0.75
${3 div 4}	O/P: 0.75
${10/0}	O/P: Infinity
${50%8}	O/P: 2
${50 mod 8}	O/P: 2
${(10==20) ? "true" : "false"}	O/P: false

guess.jsp – JSTL POST, EL and implicit object
- if else using JSTL

jstlcset.jsp

jstlcif.jsp

jstlswitch.jsp

jstlcurl.jsp

processparam.jsp
- EL accessing GET parameter

JSTL and EL operations

- jsp:useBean and scope
  - Page
    - viewpagebean.jsp
  - Request
    - CreateBeanRequestServlet
    - viewrequestbean.jsp
  - Session
    - CreateBeanRequestServlet
    - viewsessionbean.jsp
- c:forEach and jsp:useBean – populate in table
  - Page
    - foreachpage.jsp
  - Request
    - CreateBeanRequestForEachServlet
    - foreachrequest.jsp
  - Session
    - CreateBeanSessionForEachServlet
    - foreachsession.jsp