JSTL Tutorial with Examples - JSTL Core Tags

Earlier we saw how we can use JSP EL and JSP Action Tags to write JSP code like HTML but their functionality is very limited. For example, we can't loop through a collection using EL or action elements and we can't escape HTML tags to show them like text in client side.

JSP Standard Tag Library (**JSTL**) is the standard tag library that provides tags to control the JSP page behavior, iteration and control statements, internationalization tags, and SQL tags.

JSTL is part of the Java EE API and included in most servlet containers. But 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 and you can use them. You need to include these libraries in the project **WEB-INF/lib** directory. These jars are container specific, for example in Tomcat, we need to include jstl.jar and standard.jar jar files in project build path. If they are not present in the container lib directory, you should include them into your application. If you have maven project, below dependencies should be added in pom.xml file or else you will get following error in JSP pages – **eclipse Can not find the tag library descriptor for "http://java.sun.com/jsp/jstl/ core"**

```
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
</dependency>
```

Based on the JSTL functions, they are categorized into five types.

1. **Core Tags**: Core tags provide support for iteration, conditional logic, catch exception, url, forward or redirect response etc. To use JSTL core tags, we should include it in the JSP page like below.

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

   In this article, we will look into important JSTL core tags.

2. **Formatting and Localization Tags**: These tags are provided for formatting of Numbers, Dates and i18n support through locales and resource bundles. We can include these tags in JSP with below syntax:

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

3. **SQL Tags**: JSTL SQL Tags provide support for interaction with relational databases such as Oracle, MySql etc. Using SQL tags we can run database queries, we include it in JSP with below

syntax:

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

4. **XML Tags**: XML tags are used to work with XML documents such as parsing XML, transforming XML data and XPath expressions evaluation. Syntax to include XML tags in JSP page is:

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

5. **JSTL Functions Tags**: JSTL tags provide a number of functions that we can use to perform common operation, most of them are for String manipulation such as String Concatenation, Split String etc. Syntax to include JSTL functions in JSP page is:

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

Note that all the JSTL standard tags URI starts with `http://java.sun.com/jsp/jstl/` and we can use any prefix we want but it's best practice to use the prefix defined above because everybody uses them, so it will not create any confusion.

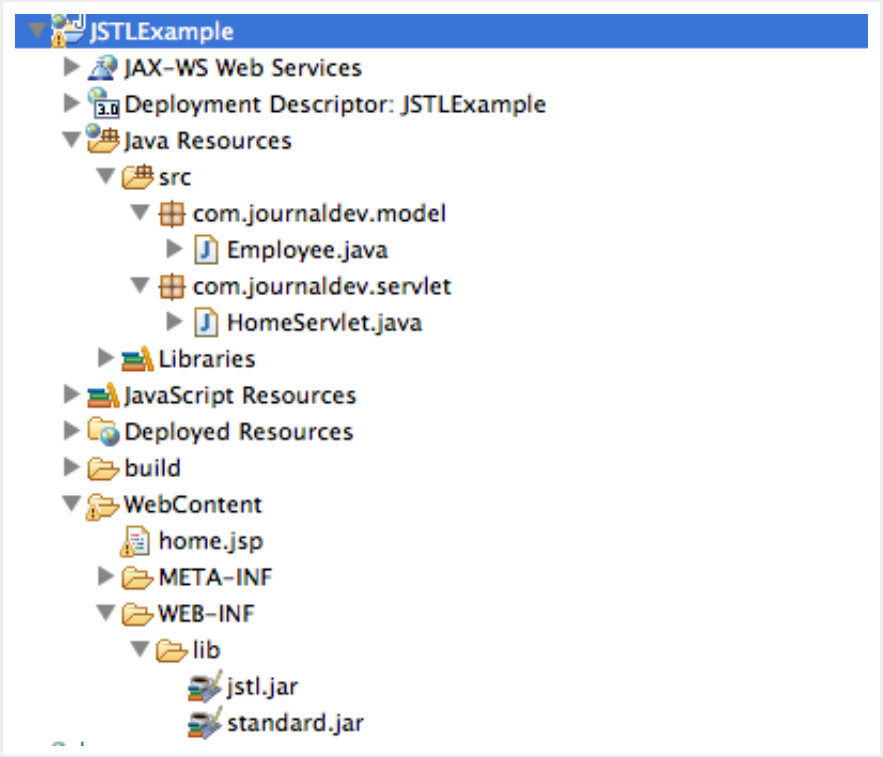
JSTL Core Tags

JSTL Core Tags are listed in the below table.

Tag	Description
<c:out>	To write something in JSP page, we can use EL also with this tag
<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
<c:forTokens>	for iteration over tokens separated by a delimiter.
<c:param>	used with <c:import> to pass parameters
<c:url>	to create a URL with optional query string parameters

Let's see some of the core tags usage with a simple web application. Our project will include a Java Bean and we will create a list of objects and set some attributes that will be used in the JSP. JSP page will show how to iterate over a collection, using conditional logic with EL and some other common usage.



Java Bean Class

Employee.java

```
1  package com.journaldev.model;
2
3  public class Employee {
4
5      private int id;
6      private String name;
7      private String role;
8      public Employee() {
9      }
10     public int getId() {
11         return id;
12     }
13     public void setId(int id) {
14         this.id = id;
15     }
16     public String getName() {
17         return name;
```

```
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getRole() {
        return role;
    }
    public void setRole(String role) {
        this.role = role;
    }

}
```

Servlet Class

HomeServlet.java

```
package com.journaldev.servlet;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.journaldev.model.Employee;

@WebServlet("/HomeServlet")
public class HomeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse resp
        List<Employee> empList = new ArrayList<Employee>();
        Employee emp1 = new Employee();
        emp1.setId(1); emp1.setName("Pankaj");emp1.setRole("Developer");
        Employee emp2 = new Employee();
        emp2.setId(2); emp2.setName("Meghna");emp2.setRole("Manager");
        empList.add(emp1);empList.add(emp2);
        request.setAttribute("empList", empList);

        request.setAttribute("htmlTagData", "<br/> creates a new line.");
        request.setAttribute("url", "http://www.journaldev.com");
        RequestDispatcher rd = getServletContext().getRequestDispatcher("/home
        rd.forward(request, response);
    }

}
```

JSP Page

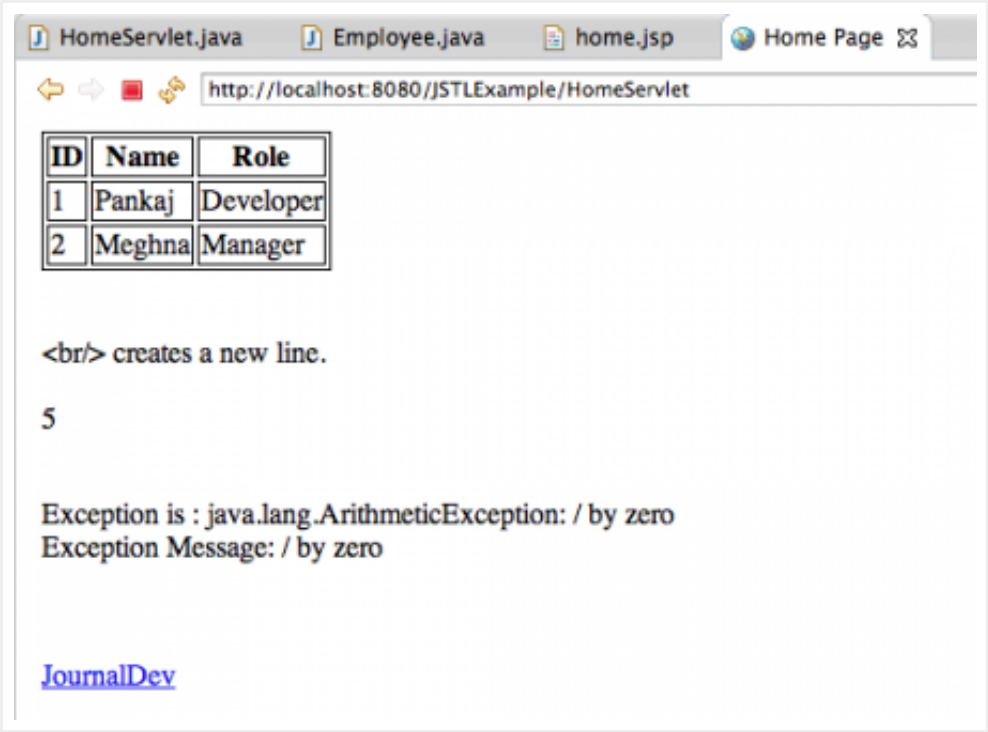
home.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Home Page</title>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<style>
table,th,td
{
border:1px solid black;
}
</style>
</head>
<body>
<%-- Using JSTL forEach and out to loop a list and display items in table --%>
<table>
<tbody>
<tr><th>ID</th><th>Name</th><th>Role</th></tr>
<c:forEach items="${requestScope.empList}" var="emp">
<tr><td><c:out value="${emp.id}"></c:out></td>
<td><c:out value="${emp.name}"></c:out></td>
<td><c:out value="${emp.role}"></c:out></td></tr>
</c:forEach>
</tbody>
</table>
<br><br>
<%-- simple c:if and c:out example with HTML escaping --%>
<c:if test="${requestScope.htmlTagData ne null }">
<c:out value="${requestScope.htmlTagData}" escapeXml="true"></c:out>
</c:if>
<br><br>
<%-- c:set example to set variable value --%>
<c:set var="id" value="5" scope="request"></c:set>
<c:out value="${requestScope.id }" ></c:out>
<br><br>
<%-- c:catch example --%>
<c:catch var ="exception">
   <% int x = 5/0;%>
</c:catch>

<c:if test = "${exception ne null}">
   <p>Exception is : ${exception} <br />
   Exception Message: ${exception.message}</p>
</c:if>
<br><br>
<%-- c:url example --%>
<a href="<c:url value="${requestScope.url }"></c:url>">JournalDev</a>
</body>
</html>
```

Now when we run application with URL `http://localhost:8080/JSTLExample/HomeServlet`, we get response as in below image.



In above example, we are using `c:catch` to catch the exception within the JSP service method, it's different from the JSP Exception Handling with error pages configurations.

Thats all for a quick roundup of JSTL and example of core tags usage, we will look into custom tags in future post.
