JSTL - JSP Standard Tag Library

The **JSP Standard Tag Library (JSTL)** is a collection of useful JSP tags that encapsulate common functionality, such as iteration, conditional logic, formatting, XML processing, and database access. It allows developers to simplify JSP development by reducing the amount of Java code written directly within JSP pages and promoting a clean, maintainable, and reusable approach for dynamic content creation.

**Key Features of JSTL**

1. **Standardization**: JSTL provides a standard set of tags that work across different JSP containers, ensuring portability and reducing dependency on custom tags.
2. **Simplicity**: JSTL tags help remove Java scriptlets (<% %>) from JSPs, making the pages more readable and maintainable.
3. **Functionality**: JSTL supports common operations like iteration, conditionals, internationalization, string manipulation, and database access.

**JSTL Tag Libraries**

JSTL is divided into several tag libraries, each focusing on a different aspect of web application development:

1. **Core Tags (c namespace)**: These tags are used for basic operations such as conditionals, loops, and variable manipulation.
2. **Formatting Tags (fmt namespace)**: These tags are for internationalization (i18n), number formatting, and date formatting.
3. **SQL Tags (sql namespace)**: These tags allow you to interact with databases.
4. **XML Tags (x namespace)**: These tags are for processing and manipulating XML data.
5. **Functions (fn namespace)**: These tags provide functions for string manipulation, such as substring operations or checking if a string contains another.

**1. JSTL Core Tags (c namespace)**

The **core tag library** provides basic functionality for iteration, conditional logic, and variable manipulation.

* **Tag Library Declaration**:

jsp

Copy code

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

**Common Core Tags**

1. **<c:out>**: Outputs the value of an expression, escaping special characters to avoid XSS attacks.

jsp

Copy code

<c:out value="${message}" />

1. **<c:set>**: Sets the value of a variable in various scopes (page, request, session, or application).

jsp

Copy code

<c:set var="username" value="John" />

1. **<c:if>**: Evaluates a condition and processes its body if the condition is true.

jsp

Copy code

<c:if test="${userLoggedIn}">

<p>Welcome, ${username}!</p>

</c:if>

1. **<c:choose>, <c:when>, <c:otherwise>**: Provides an if-else-like conditional logic structure.

jsp

Copy code

<c:choose>

<c:when test="${userRole == 'admin'}">

<p>Admin dashboard</p>

</c:when>

<c:otherwise>

<p>User dashboard</p>

</c:otherwise>

</c:choose>

1. **<c:forEach>**: Loops over a collection, such as a list or an array.

jsp

Copy code

<c:forEach var="item" items="${itemList}">

<p>${item.name}</p>

</c:forEach>

1. **<c:forTokens>**: Iterates over tokens in a string, using a specified delimiter.

jsp

Copy code

<c:forTokens items="item1,item2,item3" delims="," var="token">

<p>${token}</p>

</c:forTokens>

1. **<c:remove>**: Removes an attribute from a specific scope.

jsp

Copy code

<c:remove var="username" scope="session" />

**2. JSTL Formatting Tags (fmt namespace)**

The **formatting tag library** is used for formatting and internationalization (i18n), making it easier to handle text, dates, and numbers according to the user's locale.

* **Tag Library Declaration**:

jsp

Copy code

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

**Common Formatting Tags**

1. **<fmt:formatNumber>**: Formats a number, such as currency, percentages, or integers.

jsp

Copy code

<fmt:formatNumber value="12345.678" type="currency" />

1. **<fmt:formatDate>**: Formats a date according to a specified pattern or locale.

jsp

Copy code

<fmt:formatDate value="${currentDate}" pattern="MM/dd/yyyy" />

1. **<fmt:setLocale>**: Sets the locale for the current page.

jsp

Copy code

<fmt:setLocale value="en\_US" />

1. **<fmt:bundle>**: Loads a resource bundle for localization.

jsp

Copy code

<fmt:bundle basename="messages">

<fmt:message key="welcome" />

</fmt:bundle>

1. **<fmt:message>**: Retrieves a localized message from a resource bundle.

jsp

Copy code

<fmt:message key="greeting" />

**3. JSTL SQL Tags (sql namespace)**

The **SQL tag library** is used for interacting with databases. However, it's generally recommended to handle database interactions in the model layer rather than directly in JSP using JSTL SQL tags.

* **Tag Library Declaration**:

jsp

Copy code

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

**Common SQL Tags**

1. **<sql:setDataSource>**: Sets the database connection information.

jsp

Copy code

<sql:setDataSource var="dataSource" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/mydb"

user="root" password="password" />

1. **<sql:query>**: Executes a SQL query and stores the result in a scoped variable.

jsp

Copy code

<sql:query dataSource="${dataSource}" var="result">

SELECT \* FROM users;

</sql:query>

1. **<sql:update>**: Executes a SQL INSERT, UPDATE, or DELETE statement.

jsp

Copy code

<sql:update dataSource="${dataSource}">

INSERT INTO users (name, email) VALUES ('John', 'john@example.com');

</sql:update>

**4. JSTL XML Tags (x namespace)**

The **XML tag library** is used to work with XML data, such as reading, parsing, and transforming XML.

* **Tag Library Declaration**:

jsp

Copy code

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

**Common XML Tags**

1. **<x:parse>**: Parses XML data from a string, file, or URL.

jsp

Copy code

<x:parse var="xmlDoc" doc="${pageContext.requestURL}" />

1. **<x:out>**: Outputs an XML node or value.

jsp

Copy code

<x:out select="$xmlDoc//title" />

1. **<x:forEach>**: Iterates over XML nodes.

jsp

Copy code

<x:forEach select="$xmlDoc//book" var="book">

<p>${book.title}</p>

</x:forEach>

**5. JSTL Functions (fn namespace)**

The **functions tag library** provides utility functions for common string manipulation tasks, such as checking string length, substring operations, or replacing characters.

* **Tag Library Declaration**:

jsp

Copy code

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

**Common Functions**

1. **fn:length()**: Returns the length of a string or collection.

jsp

Copy code

${fn:length("hello")} <!-- Output: 5 -->

1. **fn:contains()**: Checks if a string contains a substring.

jsp

Copy code

${fn:contains("JSP Standard Tag Library", "Tag")} <!-- Output: true -->

1. **fn:substring()**: Returns a substring of a string.

jsp

Copy code

${fn:substring("hello world", 0, 5)} <!-- Output: hello -->

1. **fn:replace()**: Replaces occurrences of a string with another string.

jsp

Copy code

${fn:replace("JSTL is cool", "cool", "awesome")} <!-- Output: JSTL is awe

Page & PageContext

In JSP, **page** and **PageContext** are implicit objects provided to simplify web application development. They serve different purposes related to the scope of a JSP page and allow developers to interact with the page and its environment.

**1. page Object**

The **page** object is an implicit object in JSP that refers to the current instance of the servlet that the JSP is compiled into. In essence, it is the same as the this reference in Java classes. It provides access to the current JSP servlet and can be used to call methods of the servlet from within the JSP.

**Key Characteristics:**

* **Type**: java.lang.Object (in JSP), or it refers to the current servlet class (which is a subclass of HttpServlet).
* **Scope**: It is scoped to the current page, meaning it is only accessible within the JSP where it is used.

**Example:**

jsp

Copy code

<%

// Using the 'page' object to refer to the current JSP's servlet

out.println("This page is an instance of: " + page.getClass().getName());

%>

**2. PageContext Object**

The **PageContext** object is a broader and more powerful implicit object that provides access to several important JSP objects and attributes. It acts as a context for the entire JSP page and allows access to all the scopes available (page, request, session, application), as well as utility methods for managing attributes and other JSP-related operations.

**Key Characteristics:**

* **Type**: javax.servlet.jsp.PageContext.
* **Scope**: It is available for the duration of the page processing.
* **Usage**: Provides a unified way to access request, session, application, and page-level attributes.

**Common Methods of PageContext:**

1. **findAttribute(String name)**: Searches for an attribute in the page, request, session, and application scopes.

jsp

Copy code

Object attr = pageContext.findAttribute("myAttribute");

1. **getAttribute(String name, int scope)**: Retrieves an attribute from a specific scope.

jsp

Copy code

Object attr = pageContext.getAttribute("username", PageContext.SESSION\_SCOPE);

1. **setAttribute(String name, Object value, int scope)**: Sets an attribute in a specific scope.

jsp

Copy code

pageContext.setAttribute("username", "John", PageContext.SESSION\_SCOPE);

1. **getRequest()**: Returns the HttpServletRequest object associated with the current page.

jsp

Copy code

HttpServletRequest request = (HttpServletRequest) pageContext.getRequest();

1. **getSession()**: Returns the HttpSession object associated with the current request (if one exists).

jsp

Copy code

HttpSession session = pageContext.getSession();

1. **forward(String path)**: Forwards the request to another JSP, servlet, or resource.

jsp

Copy code

pageContext.forward("anotherPage.jsp");

1. **include(String path)**: Includes the content of another resource (like another JSP or servlet) into the current page.

jsp

Copy code

pageContext.include("header.jsp");

1. **removeAttribute(String name, int scope)**: Removes an attribute from a specific scope.

jsp

Copy code

pageContext.removeAttribute("username", PageContext.SESSION\_SCOPE);

**Example:**

jsp

Copy code

<%

// Setting a page attribute

pageContext.setAttribute("myMessage", "Hello, PageContext!", PageContext.PAGE\_SCOPE);

// Retrieving the page attribute

String message = (String) pageContext.getAttribute("myMessage", PageContext.PAGE\_SCOPE);

out.println("Message: " + message);

%>

**Comparison between page and PageContext**

| **Feature** | **page** | **PageContext** |
| --- | --- | --- |
| **Type** | java.lang.Object (refers to the JSP servlet) | javax.servlet.jsp.PageContext |
| **Scope** | Limited to the current JSP page | Provides access to multiple scopes |
| **Purpose** | Represents the current JSP page (servlet) | Provides a context to manage page-wide data, objects, and attributes |
| **Common Usage** | Referring to the current servlet instance | Managing attributes, forwarding requests, including resources, etc. |

ForEach

In JSP (JavaServer Pages), the **<c:forEach>** tag is part of the **JSTL (JSP Standard Tag Library)**, specifically the **core** library. It is used for iterating over a collection of objects such as arrays, lists, maps, or any other iterable data structure. This tag allows for concise and readable iteration without needing Java code (scriptlets) in your JSP.

**Tag Library Declaration**

Before using <c:forEach>, you need to include the JSTL core library in your JSP file:

jsp

Copy code

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

**Syntax of <c:forEach>**

The basic syntax of the <c:forEach> tag is:

jsp

Copy code

<c:forEach var="item" items="${collection}">

<!-- Body content -->

</c:forEach>

**Attributes of <c:forEach>:**

* **var**: Name of the variable that holds each item during iteration.
* **items**: The collection (list, array, etc.) that you want to iterate over.
* **begin** *(optional)*: Specifies the index to start the iteration.
* **end** *(optional)*: Specifies the index to end the iteration.
* **step** *(optional)*: Specifies the iteration step size (i.e., how many elements to skip between iterations).
* **varStatus** *(optional)*: Declares a variable to hold the iteration status (e.g., the current index).

**Examples of Using <c:forEach>**

**1. Simple Iteration Over a List**

jsp

Copy code

<%-- Assuming you have a List of Strings passed as an attribute in the request scope --%>

<c:forEach var="name" items="${nameList}">

<p>Name: ${name}</p>

</c:forEach>

If the nameList contains ["John", "Jane", "Doe"], the output will be:

makefile

Copy code

Name: John

Name: Jane

Name: Doe

**2. Iterating Over an Array**

jsp

Copy code

<%-- Array defined in the request attribute --%>

<%

String[] colors = {"Red", "Green", "Blue"};

request.setAttribute("colorArray", colors);

%>

<c:forEach var="color" items="${colorArray}">

<p>Color: ${color}</p>

</c:forEach>

This will print:

less

Copy code

Color: Red

Color: Green

Color: Blue

**3. Using begin, end, and step Attributes**

The begin, end, and step attributes allow you to control the iteration range and step size.

jsp

Copy code

<c:forEach var="number" items="${numberList}" begin="0" end="4" step="2">

<p>Number: ${number}</p>

</c:forEach>

If numberList contains [0, 1, 2, 3, 4, 5, 6], this will print only the numbers at index 0, 2, and 4:

javascript

Copy code

Number: 0

Number: 2

Number: 4

**4. Using varStatus to Track Iteration Status**

The varStatus attribute gives you additional information about the iteration, such as the current index, whether it's the first or last iteration, etc.

jsp

Copy code

<c:forEach var="name" items="${nameList}" varStatus="status">

<p>${status.index}: ${name} (First: ${status.first}, Last: ${status.last})</p>

</c:forEach>

For a list ["John", "Jane", "Doe"], this will print:

php

Copy code

0: John (First: true, Last: false)

1: Jane (First: false, Last: false)

2: Doe (First: false, Last: true)

**varStatus Properties:**

* **index**: Zero-based index of the current iteration.
* **count**: One-based index of the current iteration.
* **first**: Returns true if it's the first iteration, otherwise false.
* **last**: Returns true if it's the last iteration, otherwise false.

**5. Iterating Over a Map (Key-Value Pairs)**

You can also use <c:forEach> to iterate over a Map object, using the key and value.

jsp

Copy code

<%-- Assuming a Map<String, String> is set in the request attribute --%>

<%

Map<String, String> countryCapitalMap = new HashMap<>();

countryCapitalMap.put("USA", "Washington, D.C.");

countryCapitalMap.put("UK", "London");

countryCapitalMap.put("India", "New Delhi");

request.setAttribute("countryCapitalMap", countryCapitalMap);

%>

<c:forEach var="entry" items="${countryCapitalMap}">

<p>Country: ${entry.key}, Capital: ${entry.value}</p>

</c:forEach>

This will print:

yaml

Copy code

Country: USA, Capital: Washington, D.C.

Country: UK, Capital: London

Country: India, Capital: New Delhi

If, If-Else, Choose

In JSP (JavaServer Pages), you can implement conditional logic using JSTL (JavaServer Pages Standard Tag Library) tags. The key tags for conditional logic include <c:if>, <c:choose>, <c:when>, and <c:otherwise>. These tags help make your JSP code cleaner and easier to read compared to using scriptlets.

**Including the JSTL Core Library**

Before using these tags, ensure to include the JSTL core library at the top of your JSP file:

jsp

Copy code

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

**1. Using <c:if>**

The <c:if> tag allows you to conditionally display content based on a specified condition.

**Syntax:**

jsp

Copy code

<c:if test="${condition}">

<!-- Content to display if condition is true -->

</c:if>

**Example:**

jsp

Copy code

<c:if test="${userLoggedIn}">

<p>Welcome back, ${username}!</p>

</c:if>

In this example, the paragraph will be displayed only if the userLoggedIn condition evaluates to true.

**2. Using <c:if> and <c:else>**

To create an if-else structure, you can use <c:if> followed by a <c:choose> structure for more complex conditions.

**Example:**

jsp

Copy code

<c:if test="${userLoggedIn}">

<p>Welcome back, ${username}!</p>

</c:if>

<c:if test="${!userLoggedIn}">

<p>Please log in to continue.</p>

</c:if>

Alternatively, you can use <c:choose> to create a more structured if-else construct.

**3. Using <c:choose>, <c:when>, and <c:otherwise>**

The <c:choose> tag acts like a switch statement, allowing multiple conditions to be checked.

**Syntax:**

jsp

Copy code

<c:choose>

<c:when test="${condition1}">

<!-- Content if condition1 is true -->

</c:when>

<c:when test="${condition2}">

<!-- Content if condition2 is true -->

</c:when>

<c:otherwise>

<!-- Content if none of the conditions are true -->

</c:otherwise>

</c:choose>

**Example:**

jsp

Copy code

<c:choose>

<c:when test="${userLoggedIn}">

<p>Welcome back, ${username}!</p>

</c:when>

<c:when test="${!userLoggedIn && userRole == 'guest'}">

<p>Welcome, Guest! Please log in.</p>

</c:when>

<c:otherwise>

<p>Please log in to access your account.</p>

</c:otherwise>

</c:choose>

In this example:

* If userLoggedIn is true, it displays a welcome message.
* If userLoggedIn is false and the userRole is 'guest', it shows a guest message.
* If none of the conditions are met, it prompts the user to log in.