**What is JavaServer Pages ?**

JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with . A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands. Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.

**Overview of JSP Document**

JSP can be written in 2 formats:

· JSP Page - A JSP page written in an HTML format with JSP elements expressed as <% ... %> tags. This is the format which we have used so far in the tutorial.

· JSP Document - A JSP written in an XML format with JSP elements expressed as XML elements. A JSP document is JSP written in XML format and therefore must comply with the XML standard.

JSP elements included in the JSP document must comply with the XML syntax.

### ****1 Declaration****

In standard JSP page , declaration tag is used as <%! . . %> but in JSP document , declaration tag is expressed as <jsp:declaration> . . . </jsp:declaration>

### ****2 Scriptlets****

In standard JSP page , scriptlet tag is used as <% . . %> but in JSP document , scriptlets tag is expressed as <jsp:scriptlet> . . . </jsp:scriptlet>

### ****3 Expressions****

In standard JSP page , expression tag is used as <%= . . %> but in JSP document , expressions tag is expressed as <jsp:expression> . . . </jsp:expression>

### ****4 Comments****

In standard JSP page , comments are defined as <%-- . . --%> but in JSP document , expressions tag is expressed as <!-- . . . -->

### ****5 Directives****

· Include directive will be expressed as <jsp:directive.include .. /> instead of <%@include . . . %>

· Page Directive will be expressed as <jsp:directive.page .. /> instead of <%@page ... %>

· Taglib directive will be expressed as <jsp:directive.taglib .. /> instead of <%@taglib ... %>

### ****6 EL Operators****

Operators need to be used as below –

Ø < will be written as lt

Ø >will be written as gt

Ø <= will be written as Le

Ø >= will be written as Ge

Ø != will be written as Ne

## ****Overview Of****JSP Standard Tag Library (JSTL)

JSTL stands for JavaServer Pages Standard Tag Library and is a set of tags which provides functionality common to many JSP. JSTL has tags such as iterators and conditionals for handling flow control, tags for manipulating XML documents, internationalization tags, tags for accessing databases using SQL, and commonly used functions.

JSTL has a vast range of tags and because if this we need not to search for any other third party libraries (in most of the cases).

JSTL tags are divided in following groups based on the set of functionality

a) Core tag library

b) XML tag library

c) JSTL functions

d) SQL tag library

e) Internationalization tag library (Formatting Tags)

**Elements of JSP**

The elements of JSP have been described below:

**The Scriptlet**

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Following is the syntax of Scriptlet:

<% code fragment %>

**JSP Declarations**

A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.

Following is the syntax for JSP Declarations:

<%! declaration; [ declaration; ]+ ... %>

You can write the XML equivalent of the above syntax as follows:

<jsp:declaration>

code fragment

</jsp:declaration>

**JSP Expression**

A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.

Because the value of an expression is converted to a String, you can use an expression within a line of text, whether or not it is tagged with HTML, in a JSP file.

The expression element can contain any expression that is valid according to the Java Language Specification but you cannot use a semicolon to end an expression.

Following is the syntax of JSP Expression:

<%= expression %>

**JSP Directives**

A JSP directive affects the overall structure of the servlet class. It usually has the following form:

<%@ directive attribute="value" %>

|  |  |
| --- | --- |
| There are three types of directive tag: **Directive** | **Description** |
| <%@ page ... %> | Defines page-dependent attributes, such as scripting language, error page, and buffering requirements. |
| <%@ include ... %> | Includes a file during the translation phase. |
| <%@ taglib ... %> | Declares a tag library, containing custom actions, used in the page |

**JSP Actions**

JSP actions use **constructs** in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard:

<jsp:action\_name attribute="value" />

|  |  |
| --- | --- |
| Action elements are basically predefined functions. Following table lists out the available JSP Actions: **Syntax** | **Purpose** |
| jsp:include | Includes a file at the time the page is requested. |
| jsp:useBean | Finds or instantiates a JavaBean. |
| jsp:setProperty | Sets the property of a JavaBean. |
| jsp:getProperty | Inserts the property of a JavaBean into the output. |

**JSP Implicit Objects**

|  |  |
| --- | --- |
| **Objects** | **Description** |
| request | This is the **HttpServletRequest** object associated with the request. |
| response | This is the **HttpServletResponse** object associated with the response to the client. |
| out | This is the **PrintWriter** object used to send output to the client. |
| session | This is the **HttpSession** object associated with the request. |
| application | This is the **ServletContext** object associated with application context. |
| config | This is the **ServletConfig** object associated with the page. |
| pageContext | This encapsulates use of server-specific features like higher performance **JspWriters**. |
| page | This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class. |
| Exception | The **Exception** object allows the exception data to be accessed by designated JSP. |

**JSP Literals**

The JSP expression language defines the following literals:

 **Boolean:** true and false

 **Integer:** as in Java

 **Floating point:** as in Java

 **String:** with single and double quotes; " is escaped as \", ' is escaped as \', and \ is escaped as \\.

 Null: null

**The page Directive**

The **page** directive is used to provide instructions to the container. These instructions pertain to the current JSP page. You may code page directives anywhere in your JSP page. By convention, page directives are coded at the top of the JSP page.

Following is the basic syntax of the page directive:

<%@ page attribute="value" %>

**Attributes**

|  |  |
| --- | --- |
| Following table lists out the attributes associated with the page directive: **Attribute** | **Purpose** |
| buffer | Specifies a buffering model for the output stream. |
| autoFlush | Controls the behavior of the servlet output buffer. |
| contentType | Defines the character encoding scheme. |
| errorPage | Defines the URL of another JSP that reports on Java unchecked runtime exceptions. |
| isErrorPage | Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute. |
| extends | Specifies a superclass that the generated servlet must extend. |
| import | Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes. |
| info | Defines a string that can be accessed with the servlet's getServletInfo() method. |
| isThreadSafe | Defines the threading model for the generated servlet. |
| language | Defines the programming language used in the JSP page. |
| session | Specifies whether or not the JSP page participates in HTTP sessions. |
| isELIgnored | Specifies whether or not the EL expression within the JSP page will be ignored. |
| isScriptingEnabled | Determines if the scripting elements are allowed for use. |

**The buffer Attribute**

The **buffer** attribute specifies the buffering characteristics for the server output response object.

You may code a value of "**none**" to specify no buffering so that the servlet output is immediately directed to the response object or you may code a maximum buffer size in kilobytes, which directs the servlet to write to the buffer before writing to the response object.

To direct the servlet to write the output directly to the **response output object**, use the following:

<%@ page buffer="none" %>

**The contentType Attribute**

The contentType attribute sets the character encoding for the JSP page and for the generated response page. The default content type is **text/html**, which is the standard content type for HTML pages.

If you want to write out XML from your JSP, use the following page directive:

<%@ page contentType="text/xml" %>

**The errorPage Attribute**

The **errorPage** attribute tells the JSP engine which page to display if there is an error while the current page runs. The value of the errorPage attribute is a relative URL.

The following directive displays MyErrorPage.jsp when all uncaught exceptions are thrown:

<%@ page errorPage="MyErrorPage.jsp" %>

**The isErrorPage Attribute**

The **isErrorPage** attribute indicates that the current JSP can be used as the error page for another JSP.

The value of isErrorPage is either true or false. The default value of the isErrorPage attribute is false.

For example, the **handleError.jsp** sets the isErrorPage option to true because it is supposed to handle errors:

<%@ page isErrorPage="true" %>

**The extends Attribute**

The **extends** attribute specifies a superclass that the generated servlet must extend.

For example, the following directive directs the JSP translator to generate the servlet such that the servlet extends ***somePackage.SomeClass***:

<%@ page extends="somePackage.SomeClass" %>

**The import Attribute**

The **import** attribute serves the same function as, and behaves like, the Java import statement. The value for the import option is the name of the package you want to import.

To import **java.sql.\***, use the following page directive:

<%@ page import="java.sql.\*" %>

**The info Attribute**

The **info** attribute lets you provide a description of the JSP. The following is a coding example:

<%@ page info="This JSP Page Written By ZARA" %>

**The isThreadSafe Attribute**

The **isThreadSafe** option marks a page as being thread-safe. By default, all JSPs are considered thread-safe. If you set the isThreadSafe option to false, the JSP engine makes sure that only one thread at a time is executing your JSP.

The following page directive sets the **isThreadSafe** option to false:

<%@ page isThreadSafe="false" %>

**The language Attribute**

The **language** attribute indicates the programming language used in scripting the JSP page.

For example, because you usually use Java as the scripting language, your language option looks like this:

<%@ page language="java" %>

**The session Attribute**

The **session** attribute indicates whether or not the JSP page uses HTTP sessions. A value of true means that the JSP page has access to a builtin **session** object and a value of false means that the JSP page cannot access the builtin session object.

Following directive allows the JSP page to use any of the builtin object session methods such as **session.getCreationTime()** or **session.getLastAccessTime()**:

<%@ page session="true" %>

**The isELIgnored Attribute**

The **isELIgnored** attribute gives you the ability to disable the evaluation of Expression Language (EL) expressions which has been introduced in JSP 2.0.

The default value of the attribute is true, meaning that expressions, **${...}**, are evaluated as dictated by the JSP specification. If the attribute is set to false, then expressions are not evaluated but rather treated as static text.

**The include Directive**

The **include** directive is used to include a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code the ***include*** directives anywhere in your JSP page.

The general usage form of this directive is as follows:

<%@ include file="relative url" >

**JSP - Include Directive**

The **include** directive is used to include a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code *include* directives anywhere in your JSP page.

The general usage form of this directive is as follows:

<%@ include file="relative url" >

**The taglib Directive**

The JavaServer Pages API allow you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.

The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides means for identifying the custom tags in your JSP page.

The taglib directive follows the syntax given below:

<%@ taglib uri="uri" prefix="prefixOfTag" >

**JSP - The taglib Directive**

The JavaServer Pages API allows you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.

The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.

The taglib directive follows the syntax given below:

<%@ taglib uri="uri" prefix="prefixOfTag" >

# JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

## Advantage of JSTL

1. **Fast Developement** JSTL provides many tags that simplifies the JSP.
2. **Code Reusability** We can use the JSTL tags in various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

# JSTL Core Tags

The JSTL core tag provides variable support, URL management, flow control etc. The syntax used for including JSTL core library in your JSP is:

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

# JSTL Core <c:out> Tag

The <c:out> tag is similar to JSP expression tag, but it can only be used with expression. It will display the result of an expression, similar to the way < %=...% > work.

The < c:out > tag automatically escape the XML tags. Hence they aren't evaluated as actual tags.

# JSTL Core <c:import> Tag

The <c:import> is similar to jsp 'include', with an additional feature of including the content of any resource either within server or outside the server.

This tag provides all the functionality of the <include > action and it also allows the inclusion of absolute URLs.

For example: Using an import tag the content from a different FTP server and website can be accessed.

# JSTL Core <c:set> Tag

It is used to set the result of an expression evaluated in a 'scope'. The <c:set> tag is helpful because it evaluates the expression and use the result to set a value of java.util.Map or JavaBean.

This tag is similar to jsp:setProperty action tag.

# JSTL Core <c:remove> Tag

It is used for removing the specified variable from a particular scope. This action is not particularly helpful, but it can be used for ensuring that a JSP can also clean up any scope resources.

The <c:remove > tag removes the variable from either a first scope or a specified scope.

# JSTL Core <c:catch> Tag

It is used for Catches any Throwable exceptions that occurs in the body and optionally exposes it. In general it is used for error handling and to deal more easily with the problem occur in program.

The < c:catch > tag catches any exceptions that occurs in a program body.

# JSTL Core <c:if> Tag

The < c:if > tag is used for testing the condition and it display the body content, if the expression evaluated is true.

It is a simple conditional tag which is used for evaluating the body content, if the supplied condition is true.

# JSTL Core <c:choose>, <c:when>, <c:otherwise> Tag

The < c:choose > tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java**switch**statement in which we choose between a numbers of alternatives.

The <c:when > is subtag of <choose > that will include its body if the condition evaluated be 'true'.

The < c:otherwise > is also subtag of < choose > it follows &l;twhen > tags and runs only if all the prior condition evaluated is 'false'.

The c:when and c:otherwise works like if-else statement. But it must be placed inside c:choose tag.

# JSTL Core <c:forEach> Tag

The <c:for each > is an iteration tag used for repeating the nested body content for fixed number of times or over the collection.

These tag used as a good alternative for embedding a Java **while, do-while, or for**loop via a scriptlet. The < c:for each > tag is most commonly used tag because it iterates over a collection of object.

# JSTL Function Tags

The JSTL function provides a number of standard functions, most of these functions are common string manipulation functions. The syntax used for including JSTL function library in your JSP is:

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

# Expression Language (EL) in JSP

The **Expression Language** (EL) simplifies the accessibility of data stored in the Java Bean component, and other objects like request, session, application etc.

There are many implicit objects, operators and reserve words in EL.

It is the newly added feature in JSP technology version 2.0.

Implicit Objects in Expression Language (EL)

There are many implicit objects in the Expression Language. They are as follows:

Implicit Objects Usage

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

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

pageContext: it provides access to many objects request, session etc.