JSP Tutorial

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc.

A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc.

Advantage of JSP over Servlet

There are many advantages of JSP over servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet

In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

Life cycle of a JSP Page

The JSP pages follows these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (class file is loaded by the classloader)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Reqeust processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

JSP Implicit Objects

1. JSP Implicit Objects
2. out implicit object
3. Example of out implicit object

There are **9 jsp implicit objects**. These objects are *created by the web container*that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

1) JSP out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

```
PrintWriter out=response.getWriter();
```

But in JSP, you don't need to write this code.

Example of out implicit object

In this example we are simply displaying date and time.

index.jsp

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

JSP Scriptlet tag (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
<%  java source code %>
```

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

===

File: index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

File: welcome.jsp

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</form>
</body>
</html>
```

JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
<%=  statement %>
```

Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

```
<html>
<body>
<%= "welcome to jsp" %>
</body>
</html>
```

Note: Do not end your statement with semicolon in case of expression tag.

Current Time: <%= java.util.Calendar.getInstance().getTime() %>

JSP Declaration Tag

The **JSP declaration tag** is used *to declare fields and methods*.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

```
<%!  field or method declaration %>
```

Difference between JSP Scriptlet tag and Declaration tag

Jsp Scriptlet Tag	Jsp Declaration Tag
The jsp scriptlet tag can only declare variables not methods.	The jsp declaration tag can decla
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaratio method.

Example of JSP declaration tag that declares field

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag.

index.jsp

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

index.jsp

```
<html>
<body>
<%!
int cube(int n){
return n*n*n*;
}
%>
<%= "Cube of 3 is:"+cube(3) %>
</body>
</html>
```

Pass variables from servlet to jsp

Use

request.setAttribute("attributeName");

and then

```
RequestDispatcher rd = request.getRequestDispatcher("/loginError.jsp");
                    rd.include(request, response);
```

Then it will be accessible in the JSP.

JSP directives

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

- page directive
- include directive
- taglib directive

Syntax of JSP Directive

```
1. <%@ directive attribute="value" %>
```

JSP page directive

The page directive defines attributes that apply to an entire JSP page.

Syntax of JSP page directive

```
1. <%@ page attribute="value" %>
```

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored

- isThreadSafe
- autoFlush
- session
- pageEncoding
- errorPage
- isErrorPage

Example of import attribute

```
<html>
<body>

<%@ page import="java.util.Date" %>
Today is: <%= new Date() %>

</body>
</html>

```

buffer

Example of buffer attribute

```
<html>
<body>

<%@ page buffer="16kb" %>
Today is: <%= new java.util.Date() %>

</body>
</html>
```

Jsp Include Directive

Code Reusability

```
<%@ include file="header.html" %>
```

JSTL (JSP Standard Tag Library)

The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.

JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.

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.

There JSTL mainly provides 5 types of tags:

Tag Name	Description
core tags	The JSTL core tag provide variable support, URL management, flow control etc. The url for the core tag is**http://java.sun.com/jsp/jstl/core** . The prefix of core tag is **c**.
sql tags	The JSTL sql tags provide SQL support. The url for the sql tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.
xml tags	The xml sql tags provide flow control, transformation etc. The url for the xml tags is **http://java.sun.com/jsp/jstl/xml**and prefix is **x**.
internationalization tags	The internationalization tags provide support for message formatting, number and date formatting etc. The url for the internationalization tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
functions tags	The functions tags provide support for string manipulation and string length. The url for the functions tags is**http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.

For creating JSTL application, you need to load jstl.jar file.

JSTL 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

a. <C:OUT>

The <c:out> tag displays the result of an expression, similar to the way <%= %> works with a difference that <c:out> tag lets you use the simpler "." notation to access properties.

It does not work here: <%= new java.util.Date() %>

b. <C:SET>

The <c:set> tag has following attributes:

Attribute	**Description**	**Required**	**Default**
value	Information to save	No	body
target	Name of the variable whose property should be modified	No	None
property	Property to modify	No	None
var	Name of the variable to store information	No	None
scope	Scope of variable to store information	No	Page

```
<c:set var="salary" scope="session" value="${2000*2}"/>
<c:out value="${salary}"/>
```

c. C:if

The <c:if> tag evaluates an expression and displays its body content only if the expression evaluates to true.

The <c:if> tag has following attributes:

Attribute	Description	Required	Default
test	Condition to evaluate	Yes	None
var	Name of the variable to store the condition's result	No	None
scope	Scope of the variable to store the condition's result	No	page

<c:forEach>

These tags exist as a good alternative to embedding a Java **for, while,** or **do-while** loop via a scriptlet. The <c:forEach> tag is the more commonly used tag because it iterates over a collection of objects. The <c:forTokens> tag is used to break a string into tokens and iterate through each of the tokens.

Attribute:

The <c:forEach> tag has following attributes:

Attribute	Description	Required	Default
items	Information to loop over	No	None
begin	Element to start with (0 = first item, 1 = second item, ...)	No	0
end	Element to end with (0 = first item, 1 = second item, ...)	No	Last element

step	Process every step items	No	1
var	Name of the variable to expose the current item	No	None
varStatus	Name of the variable to expose the loop status	No	None

<c:param> tag has following attributes:

Attribute	**Description**	**Required**	**Default**
name	Name of the request parameter to set in the URL	Yes	None
value	Value of the request parameter to set in the URL	No	Body

If you need to pass parameters to a <c:import> tag, use the <c:url> tag to create the URL first as shown below:

```
<c:url value="/index.jsp" var="myURL">

   <c:param name="trackingId" value="1234"/>

   <c:param name="reportType" value="summary"/>

</c:url>

<c:import url="${myURL}"/>
```

Above request would pass URL as below

```
"/index.jsp?trackingId=1234;reportType=summary"
```

JSTL Functions:

JSTL includes a number of standard functions, most of which are common string manipulation functions. Following is the syntax to include JSTL Functions library in your JSP:

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

Following is the list of JSTL Functions:

Function	Description
fn:contains()	Tests if an input string contains the specified substring.
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way.
fn:endsWith()	Tests if an input string ends with the specified suffix.
fn:escapeXml()	Escapes characters that could be interpreted as XML markup.
fn:indexOf()	Returns the index withing a string of the first occurrence of a specified substring.
fn:join()	Joins all elements of an array into a string.
fn:length()	Returns the number of items in a collection, or the number of characters in a string.
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string.
fn:split()	Splits a string into an array of substrings.
fn:startsWith()	Tests if an input string starts with the specified prefix.
fn:substring()	Returns a subset of a string.
fn:substringAfter()	Returns a subset of a string following a specific substring.
fn:substringBefore()	Returns a subset of a string before a specific substring.
fn:toLowerCase()	Converts all of the characters of a string to lower case.
fn:toUpperCase()	Converts all of the characters of a string to upper case.
fn:trim()	Removes white spaces from both ends of a string.

How to format date in JSTL

<fmt:formatNumber> tag

Ensure that you have properly defined the `fmt` prefix in the JSP declarations

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

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