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.

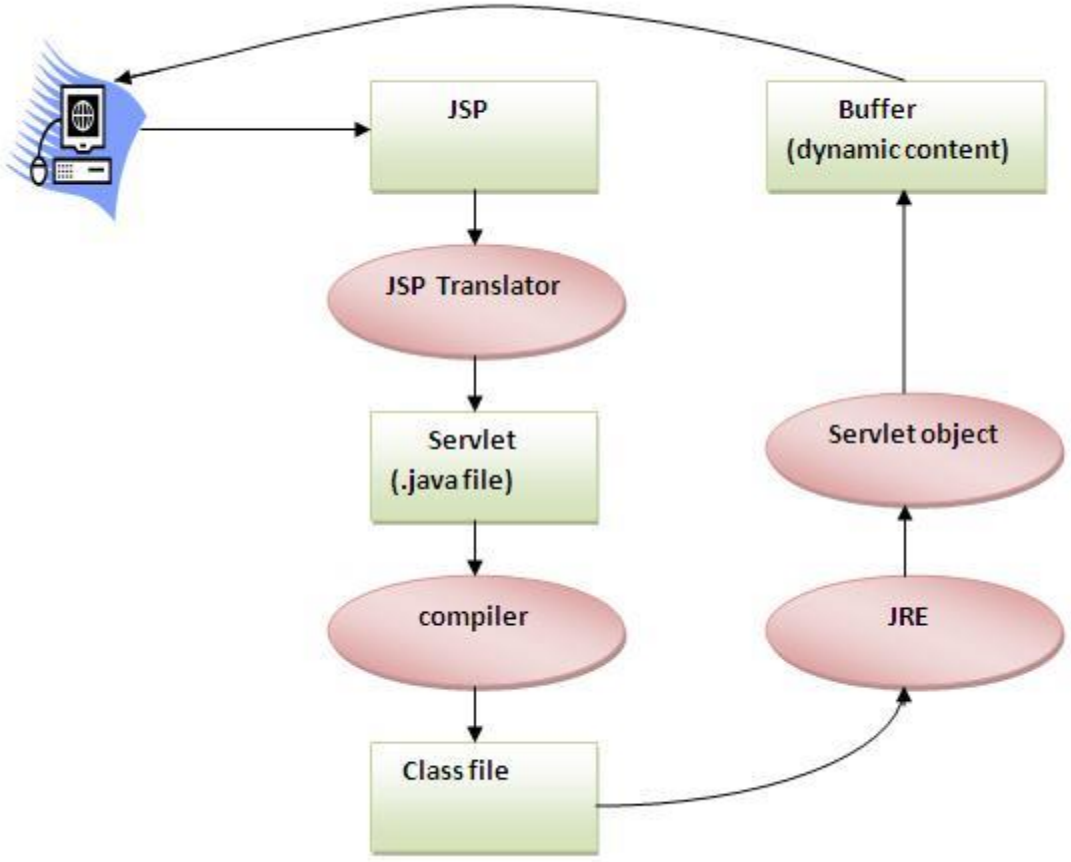
Life cycle of a JSP Page

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).

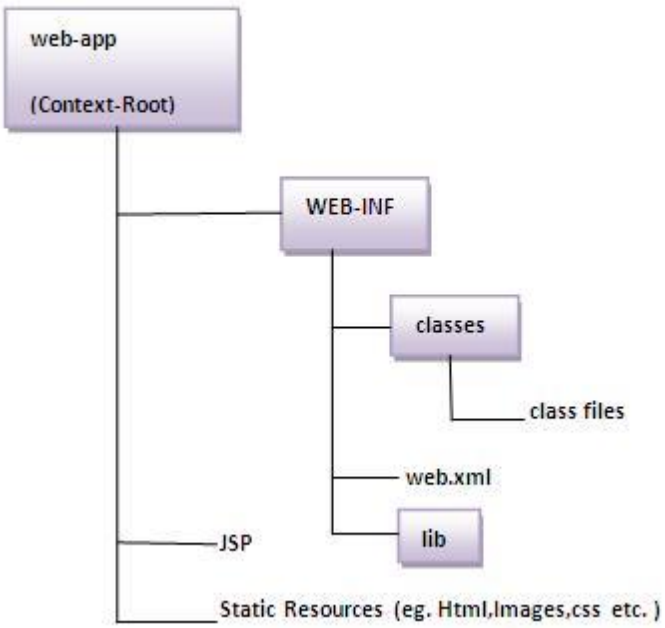
Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into servlet by the help of JSP translator. The JSP translator is a part of webserver that is responsible to translate the JSP page into servlet. Afterthat Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.

Directory structure of JSP

The directory structure of JSP page is same as servlet. We contains the jsp page outside the WEB-INF folder or in any directory.



JSP Scriptlet tag (Scripting elements)

1. Scripting elements
2. JSP scriptlet tag
3. Simple Example of JSP scriptlet tag
4. Example of JSP scriptlet tag that prints the user name

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:

1. <% java source code %>

Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

1. <html>
2. <body>
3. <% out.print("welcome to jsp"); %>
4. </body>
5. </html>

Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

File: index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

File: welcome.jsp

```
1. <html>
2. <body>
3. <%
4. String name=request.getParameter("uname");
5. out.print("welcome "+name);
6. %>
7. </form>
8. </body>
```

9. </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

1. <%= statement %>

Example of JSP expression tag

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

1. <html>
2. <body>
3. <%= "welcome to jsp" %>
4. </body>
5. </html>

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

Example of JSP expression tag that prints current time

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance

method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

index.jsp

```
1. <html>
2. <body>
3. Current Time: <%= java.util.Calendar.getInstance().g
   etTime() %>
4. </body>
5. </html>
```

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:

```
1. <%!  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 declare variables as well as methods.
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaration tag is placed outside the _jspService() 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

```
1. <html>
2. <body>
3. <%! int data=50; %>
4. <%= "Value of the variable is:"+data %>
5. </body>
6. </html>
```

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:

1. PrintWriter out=response.getWriter();

JSP directives

1. JSP directives
    1. page directive
    2. Attributes of page directive

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

1)import

The import attribute is used to import class,interface or all the members of a package.It is similar to import keyword in java class or interface.

Example of import attribute

```
1. <html>
2. <body>
3.
4. <%@ page import="java.util.Date" %>
5. Today is: <%= new Date() %>
```

6.
7. </body>
8. </html>

---

2)contentType

The contentType attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.The default value is "text/html;charset=ISO-8859-1".

Example of contentType attribute

1. <html>
2. <body>
3.
4. <%@ page contentType=application/msword %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>

---

3)extends

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

---

4)info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

Example of info attribute

```
1. <html>
2. <body>
3.
4. <%@ page info="composed by Sonoo Jaiswal" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

The web container will create a method getServletInfo() in the resulting servlet.For example:

```
1. public String getServletInfo() {
2.   return "composed by Sonoo Jaiswal";
3. }
```

---

5)buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

Example of buffer attribute

```
1. <html>
2. <body>
3.
4. <%@ page buffer="16kb" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

---

6)language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

---

7)isELIgnored

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. Expression Language is enabled by default. We see Expression Language later.

1. <%@ page isELIgnored="true" %>//Now EL will be ignored

---

8)isThreadSafe

Servlet and JSP both are multithreaded.If you want to control this behaviour of JSP page, you can use isThreadSafe attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.If you make the value of isThreadSafe attribute like:

<%@ page isThreadSafe="false" %>

The web container in such a case, will generate the servlet as:

1. public class SimplePage_jsp extends HttpJspBase
2. implements SingleThreadModel{
3. .....
4. }

---

9)errorPage

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

Example of errorPage attribute

```
1. //index.jsp
2. <html>
3. <body>
4.
5. <%@ page errorPage="myerrorpage.jsp" %>
6.
7.  <%= 100/0 %>
8.
9. </body>
10.      </html>
```

Jsp Include Directive

1. Include directive
2. Advantage of Include directive
3. Example of include directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

Advantage of Include directive

Code Reusability

Syntax of include directive

1. <%@ include file="resourceName" %>

Example of include directive

In this example, we are including the content of the header.html file. To run this example you must create an header.html file.

```
1. <html>
2. <body>
3.
4. <%@ include file="header.html" %>
5.
6. Today is: <%= java.util.Calendar.getInstance().getTi
   me() %>
7.
8. </body>
9. </html>
```

Note: The include directive includes the original content, so the actual page size grows at runtime.

JSP Taglib directive

1. JSP Taglib directive

2. Example of JSP Taglib directive

The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

Syntax JSP Taglib directive

1. <%@ taglib uri="uriofthetaglibrary" prefix="prefixof taglibrary" %>

Example of JSP Taglib directive

In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

```
1. <html>
2. <body>
3.
4. <%@ taglib uri="http://www.javatpoint.com/tags" pr
   efix="mytag" %>
5.
6. <mytag:currentDate/>
7.
8. </body>
```

9. </html>

Execution process of JSP Applications:

Following link

https://www.javatpoint.com/creating-jsp-in-eclipse-ide

JSP Action Tags

There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks.

The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are given below.

JSP Action Tags	Description
jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.

jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

The jsp:useBean, jsp:setProperty and jsp:getProperty tags are used for bean development. So we will see these tags in bean developement.

---

jsp:forward action tag

The jsp:forward action tag is used to forward the request to another resource it may be jsp, html or another resource.

Syntax of jsp:forward action tag without parameter

1. <jsp:forward page="relativeURL | <%= expression %>" />

Syntax of jsp:forward action tag with parameter

1. <jsp:forward page="relativeURL | <%= expression %>">
2. <jsp:param name="parametername" value="parameterrvalue | <%=expression%>" />
3. </jsp:forward>

jsp:useBean action tag

1. jsp:useBean action tag
2. Syntax of jsp:useBean action tag
3. Attributes and Usage of jsp:useBean action tag
4. Simple example of jsp:useBean action tag

The jsp:useBean action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

Syntax of jsp:useBean action tag

1. <jsp:useBean id= "instanceName" scope= "page | request | session | application"
2. class= "packageName.className" type= "packageName.className"
3. beanName="packageName.className | <%= expression >" >
4. </jsp:useBean>

Attributes and Usage of jsp:useBean action tag

1. **id:** is used to identify the bean in the specified scope.
2. **scope:** represents the scope of the bean. It may be page, request, session or application. The default scope is page.

- **page:** specifies that you can use this bean within the JSP page. The default scope is page.
- **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
- **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
- **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope than session.

3. **class:** instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.
4. **type:** provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.
5. **beanName:** instantiates the bean using the java.beans.Beans.instantiate() method.

jsp:include action tag

1. jsp:include action tag
2. Syntax of jsp:include action tag
3. Example of jsp:include action tag without parameter

The **jsp:include action tag** is used to include the content of another resource it may be jsp, html or servlet.

The jsp include action tag includes the resource at request time so it is **better for dynamic pages** because there might be changes in future.

The jsp:include tag can be used to include static as well as dynamic pages.

Advantage of jsp:include action tag

Code reusability : We can use a page many times such as including header and footer pages in all pages. So it saves a lot of time.

Difference between jsp include directive and include action

JSP include directive	JSP include action
includes resource at translation time.	includes resource at request time.
better for static pages.	better for dynamic pages.
includes the original content in the generated servlet.	calls the include method.

Syntax of jsp:include action tag without parameter

1. <jsp:include page="relativeURL | <%= expression %>" />

Syntax of jsp:include action tag with parameter

1. <jsp:include page="relativeURL | <%= expression %>">
2. <jsp:param name="parametername" value="parametervalue | <%=expression%>" />
3. </jsp:include>

jsp:setProperty and jsp:getProperty action tags

The setProperty and getProperty action tags are used for developing web application with Java Bean. In web devlopment, bean class is mostly used because it is a reusable software component that represents data.

The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

Syntax of jsp:setProperty action tag

1. <jsp:setProperty name="instanceOfBean" property="*" |
2. property="propertyName" param="parameterName" |

3. property="propertyName" value="{ string | <%= exp
   ression %>}"
4. />

Example of jsp:setProperty action tag if you have to set all the values of incoming request in the bean

1. <jsp:setProperty name="bean" property="*" />

Example of jsp:setProperty action tag if you have to set value of the incoming specific property

1. <jsp:setProperty name="bean" property="username"
   />

Example of jsp:setProperty action tag if you have to set a specific value in the property

1. <jsp:setProperty name="bean" property="username"
   value="Kumar" />

---

jsp:getProperty action tag

The jsp:getProperty action tag returns the value of the property.

Syntax of jsp:getProperty action tag

1. <jsp:getProperty name="instanceOfBean" property="
   propertyName" />

Simple example of jsp:getProperty action tag

1. <jsp:getProperty name="obj" property="name" />
