**JSP Standard Action Tags: include, useBean, forward, param**

**What is JSP Action?**

* JSP actions use the construct in XML syntax to control the behavior of the servlet engine.
* We can dynamically insert a file, reuse the beans components, forward user to another page, etc. through JSP Actions like include and forward.
* Unlike directives, actions are re-evaluated each time the page is accessed.

**Syntax:**

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

In this tutorial, you will learn about Actions. Actions are used for controlling the behavior of servlet engine.

**How many standard Action Tags are available in JSP?**

There are 11 types of Standard Action Tags as following:

* jsp:useBean
* jsp:include
* jsp:setProperty
* jsp:getProperty
* jsp:forward
* jsp:plugin
* jsp:attribute
* jsp:body
* jsp:text
* jsp:param
* jsp:attribute
* jsp:output

1. **jsp:useBean:**

* This action name is used when we want to use beans in the JSP page.
* With this tag, we can easily invoke a bean.

**Syntax of jsp: UseBean**:

<jsp:useBean id="" class="" />

Here it specifies the identifier for this bean and class is full path of the bean class

**Example:**

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Action JSP1</title>
8. </head>
9. <body>
10. <jsp:useBean id="name" class="demotest.DemoClass">
11. </body>
12. </html>

**Explanation of the code:**

**Code Line 10:** In the above code we use "bean id" and "class path" of the bean.

1. **jsp:include**

* It also used to insert a jsp file into another file, just like include directive.
* It is added during request processing phase

**Syntax of jsp:include**

<jsp:include page="page URL" flush="true/false">

**Example:**

Action\_jsp2 (Code Line 10) we are including a date.jsp file

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Date Guru JSP</title>
8. </head>
9. <body>
10. <jsp:include page="date.jsp" flush="true" />
11. </body>
12. </html>

Date.jsp

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Insert title here</title>
8. </head>
9. <body>
10. <p>
11. Today's date: <%= {new java.util.Date()).toLocaleString()%>
12. </p>
13. </body>
14. </html>

**Explanation of the code:**

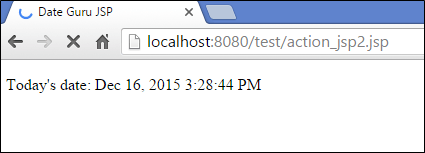
Action\_jsp2.jsp

**Code Line 10:** In the first file we are including the date.jsp file in action\_jsp2.jsp

Date.jsp:

**Code Line 11:** We are printing today's date in code line 11 in date.jsp

When you execute the code following is the output.

[](https://www.guru99.com/images/jsp/022716_0953_JSPActionsJ31.png)

**Output:**

* It displays today's date with time as date file is included in the main jsp

1. **jsp:setProperty**

* This property is used to set the property of the bean.
* We need to define a bean before setting the property

**Syntax:**

<jsp:setproperty name="" property="" >

Here, the name defines the bean whose property is set and property which we want to set.

Also, we can set value and param attribute.

Here value is not mandatory, and it defines the value which is assigned to the property.

Here param is the name of the request parameter using which value can be fetched.

The example of setproperty will be demonstrated below with getproperty

1. **jsp:getProperty**

* This property is used to get the property of the bean.
* It converts into a string and finally inserts into the output.

**Syntax:**

<jsp:getAttribute name="" property="" >

Here, the name of the bean from which the property has to be retrieved and bean should be defined. The property attribute is the name of the bean property to be retrieved.

**Example of setProperty and getProperty:**

**TestBean.java:**

1. package demotest;
3. import java.iO.Serializable;
5. public class TestBean implements Serializable{
7. private String msg = "null";
9. public String getMsg() {
10. return msg;
11. }
13. public void setMsg(String msg) {
14. this.msg = msg;
15. }
16. }

**Action\_jsp3.jsp**

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Guru Action 3</title>
8. </head>
9. <body>
10. <jsp:useBean id="GuruTest" class="demotest.TestBean" />
11. <jsp:setProperty name="GuruTest" property="msg" value="GuruTutorial" />
12. <jsp:getProperty name="GuruTest" property="msg" />
13. </body>
14. </html>

**Explanation of the code:**

TestBean.java:

**Code Line 5**: TheTestBean is implementing the serializable class. It is a bean class with getters setters in the code.

**Code Line 7**: Here we are taking private string variable msg as "null"

**Code Line 9-14**: Here we are using getters and setters of variable "msg".

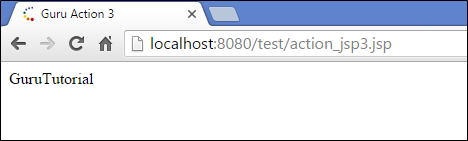
Action\_jsp3.jsp

**Code Line 10:** Here we are using "useBean" tag, where it specifies the bean i.e TestBean which has to be used in this jsp class

**Code Line 11:** Here we are setting the value for the property msg for bean TestBean as "GuruTutorial."

CodeLine12: Here using getProperty, we are getting the value of property msg for bean TestBean i.e GuruTutorial which is there in the output

When you execute the above code you get the following output:

[](https://www.guru99.com/images/jsp/022716_0953_JSPActionsJ34.png)

**Output:**

In this example, using TestBean we are trying to set the property "gurutest" using setProperty and get the value of property using getProperty as "GuruTutorial"

1. **jsp:forward:**

It is used to forward the request to another jsp or any static page.

Here the request can be forwarded with no parameters or with parameters.

**Syntax:**

<jsp:forward page="value">

Here value represents where the request has to be forwarded.

**Example:**

Action\_jsp41.jsp

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Guru Action JSP1</title>
8. </head>
9. <body>
10. <jsp:forward page="jsp\_action\_42.jsp" />
11. </body>
12. </html>

Jsp\_action\_42.jsp

1. Guru Action JSP2</title>
2. </head>
3. <body>
4. <a>This is after forward page</a>
5. </body>
6. </html>

**Explanation of the code**

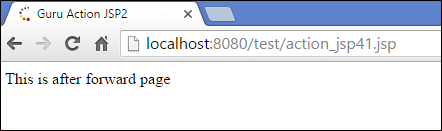
Action\_jsp41.jsp

**Code Line 10:** Here we are using forward JSP Action to forward the request to the page mentioned in the attribute, i.e., jsp\_action\_42.jsp

Jsp\_action\_42.jsp

**Code Line 10:** Once we call action\_jsp41.jsp, the request gets forwarded to this page, and we get the output as "This is after forward page."

When we execute the above code, we get the following output

[](https://www.guru99.com/images/jsp/022716_0953_JSPActionsJ37.png)

**Output:**

We call action\_jsp41.jsp but the request gets forwarded to jsp\_action\_42.jsp, and we get the output from that page as "This is after forward page".

1. **jsp:plugin**

* It is used to introduce[Java](https://www.guru99.com/java-tutorial.html)components into jsp, i.e., the java components can be either an applet or bean.
* It detects the browser and adds <object> or <embed> tags into the file

**Syntax:**

<jsp:plugin type="applet/bean" code="objectcode" codebase="objectcodebase">

* Here the type specifies either an object or a bean
* Code specifies class name of applet or bean
* Code base contains the base URL that contains files of classes

1. **jsp:param**

* This is child object of the plugin object described above
* It must contain one or more actions to provide additional parameters.

**Syntax:**

<jsp:params>

<jsp:param name="val" value="val"/ >

</jsp:params>

Example of plugin and param

Action\_jsp5.jsp

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Action guru jsp5</title>
8. </head>
9. <body>
10. <jsp:plugin type="bean" code="Student.class" codebase="demotest.Student">
11. <jsp:params>
12. <jsp:param name="id" value="5" />
13. <jsp:param name="name" value="guru" />
14. </jsp:params>
15. </jsp:plugin>
16. </body>
17. </html>

Student.java

1. package demotest;
3. import java.io.Serializable;
5. public class Student implements Serializable {
7. public String getName () {
8. return name;
9. }
10. public void setName (String name) {
11. this.name = name;
12. }
13. public int getId() {
14. return id;
15. }
16. public void setId (int id) {
17. this.id = id;
18. }
19. private String name = "null";
20. private int id = 0;
22. }

**Explanation of the code:**

Action\_jsp5.jsp

**Code Line 10:** Here we are taking jsp: plugin object where we are taking three attributes

* Type – in this case it is bean
* Code- name of the file
* Codebase – path with the package name

**Code Line 11-14:**Here we are taking jsp: params object under which there is a child param object with the attributes of name and value, and we are setting the values of id and name in this attributes.

Student.java

**Code 7- 17:**We are using getters and setters for variables id and name

**Code 19-20:**we are initializing variables id and name.

Here we will get output in the case when the set values of param will be used in Student Bean. In this case, we won't have any output as we are just setting and getting values of param but not printing it anywhere.

1. **jsp:body**

* This tag is used to define the XML dynamically i.e., the elements can generate during request time than compilation time.
* It actually defines the XML, which is generated dynamically element body.

**Syntax:**

<jsp:body></jsp:body>

Here we write XML body tag within this tags

1. **jsp:attribute**

* This tag is used to define the XML dynamically i.e. the elements can be generated during request time than compilation time
* It actually defines the attribute of XML which will be generated dynamically.

**Syntax:**

<jsp:attribute></jsp:attribute>

Here we write attribute tag of XML.

**Example of body and attribute:**

Action\_jsp6.jsp

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Action Guru JSP6</title>
8. </head>
9. <body>
10. <jsp:element name="GuruXMLElement">
11. <jsp:attribute name="GuruXMLattribute">
12. Value
13. </jsp:attribute>
14. <jsp:body>Guru XML</jsp:body>
15. </jsp:element>
16. </body>
17. </html>

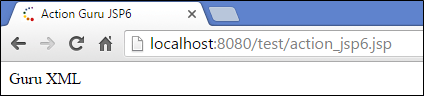
**Explanation of the code:**

**Code Line 10:**Here we are defining element, which dynamically generated as XML, and its name will be GuruXMLElement

**Code Line 11-13:**Here we are defining an attribute which will XML attribute of the dynamically generated XML.

**Code Line 14:**Here we have body action where we are writing the XML body which will be generated in dynamically XML.

When you execute the above code, you get the following output:

[](https://www.guru99.com/images/jsp/022716_0953_JSPActionsJ41.png)

**Output:**

Here we get the output from the body tag of generated XML.

1. **jsp:text**

* It is used to template text in JSP pages.
* Its body does not contain any other elements, and it contains only text and EL expressions.

**Syntax:**

<jsp:text>template text</jsp:text>

Here template text refers to only template text (which can be any generic text which needs to be printed on jsp ) or any EL expression.

**Example:**

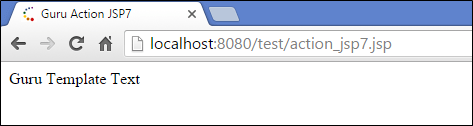
Action\_jsp7.jsp

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Guru Action JSP7</title>
8. </head>
9. <body>
10. <jsp:text>Guru Template Text</jsp:text>
11. </body>
12. </html>

**Explanation of the code:**

**Code Line 10:**Here we are taking text object to print the template text

When you execute the above code, you get the following output

[](https://www.guru99.com/images/jsp/022716_0953_JSPActionsJ43.png)

**Output:**

We are getting Guru Template Text, which is placed within text action objects.

**11) output**:

* It specifies the XML declaration or the DOCTYPE declaration of jsp
* The XML declaration and DOCTYPE are declared by the output

**Syntax:**

<jsp:output doctype-root-element="" doctype-system="">

Here, doctype-root-element indicates the root element of XML document in DOCTYPE.

Doctype-system indicates doctype which is generated in output and gives system literal

**Example:**

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Action Guru JSP8</title>
8. </head>
9. <body>
10. <jsp:output doctype-root-element="html PUBLIC" doctype-system="http://www.w3.org/TR/html4/loose.dtd"/>
11. </body>
12. </html>

**Explanation of the code:**

**Code Line 10**: Here we are using output action object to generate a DOCTYPE, and internally it will be generated in this format:

<!DOCTYPE html "http://www.w3.org/TR/html4/loose.dtd">

There won't be any output for this as this will be generated internally.