Scripting Element	Example
Comment	<%-- comment --%>
Directive	<%@ directive %>
Declaration	<%! declarations %>
Scriptlet	<% scriplets %>
Expression	<%= expression %>

Types of JSP Scripting Elements

JSP scripting elements let you insert Java code into the servlet that will be generated from the JSP page. There are three forms:

1- Expressions of the form, which are evaluated and inserted into the servlet's output.
2- Scriptlets of the form, which are inserted into the servlet's _jspService method (called by service).
3- Declarations of the form, which are inserted into the body of the servlet class, outside any existing methods.

JSP Expressions

A JSP expression is used to insert values directly into the output. It has the following form:

<% = Java Expression %>.

The expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at runtime (when the page is requested) and thus has full access to information about the request.

For example, the following shows the date/time that the page was requested. Current time: <% = java.util.Date() %>

Predefined Variables

To simplify these expressions, you can use a number of predefined variables (or "implicit objects"). The system simply tells you what names it will use for the local variables in _jspService (the method that replaces doGet in servlets that result from JSP pages). These implicit objects are these:

1- request, the HttpServletRequest.

2- response, the HttpServletResponse.

3- session, the HttpSession associated with the request (unless disabled with the session attribute of the page directive—see Section 12.4).

4- out, the Writer (a buffered version of type JspWriter) used to send output to the client.

5- application, the ServletContext. This is a data structure shared by all servlets and JSP pages in the Web application and is good for storing shared data.

```
Host Name : <% = request.getRemoteHost() %>
```

JSP/Servlet Correspondence

JSP expressions basically become print (or write) statements in the servlet that results from the JSP page. Whereas regular HTML becomes print statements with double quotes around the text, JSP expressions become print statements with no double quotes. Instead of being placed in the doGet method, these print statements are placed in a new method called _jspService() that is called by service for both GET and POST requests.

FileName: LuckyNumber.jsp

```
<html>
  <body>
      Your Luck Number is: <% = Math.random() %>
  <body>
<html>
```

FileName: LuckyNumber.java

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {

        response.setContentType("text/html");
```

```
            HttpSession session = request.getSession();
            JspWriter out = response.getWriter();

            out.println ("<html><body>");
            out.println ("Your Luck Number is :");
            out.println (Math.random());
            out.println ("<body><html>");
}
```

Servlet and Corresponding JSP

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ThreeParams extends HttpServlet
{
public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {

  response.setContentType("text/html");
  PrintWriter out = response.getWriter(); String
  title = "Reading Request Parameters";

  out.println("<html><head><title>" + title + "</title></head> \n");
                "<Body bgcolor = \"#F4F5E3\" > \n <UL> \n" +
                "<LI> User Name:" +
                request.getParameter("userName") + " \n" +
                "<LI> Department:" +
                request.getParameter("deptName") + " \n" +
                "</UL> \n </Body> \ n <html>");
 }
}
```

Corresponding JSP

```
<html>
     <head>
          <title>Reading Request Parameters </title>
     </head>
 <Body bgcolor = #F4F5E3 >
  <UL>
     <LI> User Name: <%= request.getParameter("userName") %>
     <LI> Department: <%= request.getParameter("deptName") %>
  </UL>
 </Body>
```

```
<html>
```

JSP scriptlet tag

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

<% java source code %>

Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

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

```
<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>
        <form>
```

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

JSP 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
<%@ 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.

```
<html>
<body>
        <%@ include file="Header.html" %>
        Today is: <%= java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```

<%@ include file="filename" %> is the JSP include directive.

At JSP page translation time, the content of the file given in the include directive is 'pasted' as it is, in the place where the JSP include directive is used. Then the source JSP page is converted into a java Servlet class. The included file can be a static resource or a JSP page. Generally JSP include directive is used to include header banners and footers.

The JSP compilation procedure is that, the source JSP page gets compiled only if that page has changed. If there is a change in the included JSP file, the source JSP file will not be compiled and therefore the modification will not get reflected in the output.

<jsp:include page="relativeURL" /> is the JSP include action element.

When the included JSP page is called, both the request and response objects are passed as parameters.

If there is a need to pass additional parameters, then <jsp:param> element can be used. If the resource is static, its content is inserted into the calling JSP file, since there is no processing needed.

Example: Passing Parameter in <jsp:include>

```
<html> <head> <title>JSP include with parameters</title></head>
    <body>
        <jsp:include page="InstituteList.jsp" >
            <jsp:param name="governingBody" value="Government" />
            <jsp:param name="instituteType" value="Polytechnic" />
            <jsp:param name="insttituteLocation" value="Lohaghat" />
        </jsp:include>
    </body>
</html>
```

JSP Action Tags Description

1. jsp:forward	Forwards the request & response to other resource.
2. jsp:include	Includes another resource.
3. jsp:useBean	Creates or locates bean object.
4. jsp:setProperty	Sets the value of property in bean object.
5. jsp:getProperty	Prints the value of property of the bean.
6. jsp:plugin	Embeds another components such as applet.
7. jsp:param	Sets the parameter values used in forward and include.

Java Beans

A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications. Following are the unique characteristics that distinguish a JavaBean from other Java classes –

1. It provides a default, no-argument constructor.
2. It should be serializable and that which can implement the Serializable interface.
3. It may have a number of properties which can be read or written.
4. It may have a number of "getter" and "setter" methods for the properties.

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 then it instantiates the bean.

Syntax of jsp:useBean action tag