JSP: In Java, **JSP** stands for **Java Server Pages**. It is a server-side technology which is used for creating web applications. It is used to create dynamic web content. JSP consists of both HTML tags and JSP tags. In this, JSP tags are used to insert JAVA code into HTML pages. It is an advanced version of **Servlet** Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages. In this, Java code can be inserted in HTML/ XML pages or both. JSP is first converted into a servlet by the JSP container before processing the client's request. JSP has various features like JSP Expressions, JSP tags, JSP Expression Language, etc.

How is JSP more advantageous than Servlet?

- They are easy to maintain.
- No recompilation or redeployment is required.
- Less coding is required in JSP.
- JSP has access to the entire API of JAVA.
- JSP is an extended version of Servlet.

Explain JSP Elements:

We will learn about the various elements available in JSP with suitable examples. In JSP elements can be divided into 4 different types.

These are:

- Expression
- Scriplets
- Directives
- Declarations

Following steps are involved in the JSP life cycle:

1. **Translation of JSP page to Servlet**
2. **Compilation of JSP page(Compilation of JSP into test.java)**
3. **Classloading (test.java to test.class)**
4. **Instantiation(Object of the generated Servlet is created)**
5. **Initialization(jspInit() method is invoked by the container)**
6. **Request processing(_jspService()is invoked by the container)**
7. **JSP Cleanup (jspDestroy() method is invoked by the container)**

JSP - Directives

In this chapter, we will discuss Directives in JSP. These directives provide directions and instructions to the container, telling it how to handle certain aspects of the JSP processing.

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

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

Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.

The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

There are three types of directive tag −

S.No.	Directive & Description
1	**<%@ page ... %>** Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
2	**<%@ include ... %>** Includes a file during the translation phase.
3	**<%@ taglib ... %>** Declares a tag library, containing custom actions, used in the page

JSP - 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" %>
```

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

```
<jsp:directive.page attribute = "value" />
```

Attributes:

Following table lists out the attributes associated with the page directive −

S.No.	Attribute & Purpose
1	**buffer** Specifies a buffering model for the output stream.
2	**autoFlush** Controls the behavior of the servlet output buffer.

3	**contentType** Defines the character encoding scheme.
4	**errorPage** Defines the URL of another JSP that reports on Java unchecked runtime exceptions.
5	**isErrorPage** Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.
6	**extends** Specifies a superclass that the generated servlet must extend.
7	**import** Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes.
8	**info** Defines a string that can be accessed with the servlet's **getServletInfo()** method.
9	**isThreadSafe** Defines the threading model for the generated servlet.
10	**language** Defines the programming language used in the JSP page.
11	**session** Specifies whether or not the JSP page participates in HTTP sessions
12	**isELIgnored** Specifies whether or not the EL expression within the JSP page will be ignored.
13	**isScriptingEnabled** Determines if the scripting elements are allowed for use.

Check for more details related to all the above attributes at Page Directive.

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" >
```

The filename in the include directive is actually a relative URL. If you just specify a filename with no associated path, the JSP compiler assumes that the file is in the same directory as your JSP.

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

```
<jsp:directive.include file = "relative url" />
```

For more details related to include directive, check the Include Directive.

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" >
```

Here, the **uri** attribute value resolves to a location the container understands and the **prefix** attribute informs a container what bits of markup are custom actions.

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

```
<jsp:directive.taglib uri = "uri" prefix = "prefixOfTag" />
```

For more details related to the taglib directive, check the Taglib Directive.

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

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

Syntax of jsp:forward action tag with parameter

```
<jsp:forward page="relativeURL | <%= expression %>">
<jsp:param name="parametername" value="parametervalue | <%=expression%>" />
</jsp:forward>
```

Example of jsp:forward action tag without parameter

In this example, we are simply forwarding the request to the printdate.jsp file.

index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" />
</body>
</html>
```

printdate.jsp

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

download this example

Example of jsp:forward action tag with parameter

In this example, we are forwarding the request to the printdate.jsp file with parameter and printdate.jsp file prints the parameter value with date and time.

index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" >
<jsp:param name="name" value="javatpoint.com" />
</jsp:forward>

</body>
</html>
```

printdate.jsp

```
<html>
<body>

```

```
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
<%= request.getParameter("name") %>

</body>
</html>

```