JavaServer Pages often serve the same purpose as programs implemented using the **Common Gateway Interface (CGI)**. But JSP offers several advantages in comparison with the CGI.

* Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
* JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
* JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including **JDBC, JNDI, EJB, JAXP,** etc.
* JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

The JSP container is responsible for intercepting requests for JSP pages.

A JSP life cycle is defined as the process from its creation till the destruction. This is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

Paths Followed By JSP

The following are the paths followed by a JSP −

* Compilation
* Initialization
* Execution
* Cleanup

The elements of JSP have been described below −

### **The Scriptlet**

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Following is the syntax of Scriptlet −

<% code fragment %>

<html>

<head><title>Hello World</title></head>

<body>

Hello World!<br/>

<%

out.println("Your IP address is " + request.getRemoteAddr());

%>

</body>

</html>

### **JSP Declarations**

A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.

Following is the syntax for JSP Declarations −

<%! declaration; [ declaration; ]+ ... %>

Following is an example for JSP Declarations −

<%! int i = 0; %>

<%! int a, b, c; %>

<%! Circle a = new Circle(2.0); %>

### **JSP Expression**

A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.

The expression element can contain any expression that is valid according to the Java Language Specification but you cannot use a semicolon to end an expression.

Following is the syntax of JSP Expression −

<%= expression %>

Following example shows a JSP Expression −

<html>

<head><title>A Comment Test</title></head>

<body>

<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>

</body>

</html>

### **JSP Comments**

JSP comment marks text or statements that the JSP container should ignore. A JSP comment is useful when you want to hide or "comment out", a part of your JSP page.

Following is the syntax of the JSP comments −

<%-- This is JSP comment --%>

Following example shows the JSP Comments −

<html>

<head><title>A Comment Test</title></head>

<body>

<h2>A Test of Comments</h2>

<%-- This comment will not be visible in the page source --%>

</body>

</html>

### **JSP Directives**

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

<%@ directive attribute="value" %>

### **JSP Actions**

JSP actions use **constructs** in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard −

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

### **JSP Literals**

The JSP expression language defines the following literals −

* **Boolean** − true and false
* **Integer** − as in Java
* **Floating point** − as in Java
* **String** − with single and double quotes; " is escaped as \", ' is escaped as \', and \ is escaped as \\.
* **Null** − null
* There are two attributes that are common to all Action elements: the **id** attribute and the **scope** attribute.

### **Id attribute**

* The id attribute uniquely identifies the Action element, and allows the action to be referenced inside the JSP page. If the Action creates an instance of an object, the id value can be used to reference it through the implicit object PageContext.

### **Scope attribute**

* This attribute identifies the lifecycle of the Action element. The id attribute and the scope attribute are directly related, as the scope attribute determines the lifespan of the object associated with the id. The scope attribute has four possible values: **(a) page, (b)request, (c)session**, and **(d) application**.

## The <jsp:include> Action

This action lets you insert files into the page being generated. The syntax looks like this −

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

## The Methods in Form Processing

Let us now discuss the methods in Form Processing.

### **GET method**

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character as follows −

http://www.test.com/hello?key1=value1&key2=value2

if you have password or other sensitive information to pass to the server.

The GET method has size limitation: **only 1024 characters can be in a request string**.

<html>

<head>

<title>Using GET Method to Read Form Data</title>

</head>

<body>

<h1>Using GET Method to Read Form Data</h1>

<ul>

<li><p><b>First Name:</b>

<%= request.getParameter("first\_name")%>

</p></li>

<li><p><b>Last Name:</b>

<%= request.getParameter("last\_name")%>

</p></li>

</ul>

</body>

</html>

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

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 |

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

## Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

## JSTL Tags

There JSTL mainly provides five types of tags:

|  |  |
| --- | --- |
| **Tag Name** | **Description** |
| [Core tags](https://www.javatpoint.com/jstl-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**. |
| [Function tags](https://www.javatpoint.com/jstl-function-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**. |
| [Formatting tags](https://www.javatpoint.com/jstl-formatting-tags) | The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**. |
| [XML tags](https://www.javatpoint.com/jstl-xml-tags) | The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**. |
| [SQL tags](https://www.javatpoint.com/jstl-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**. |