# JSP - Syntax

In this chapter, we will discuss Syntax in JSP. We will understand the basic use of simple syntax (i.e, elements) involved with JSP development.

## Elements of JSP

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 %>
```

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

```
<jsp:scriptlet>
  code fragment
</jsp:scriptlet>
```

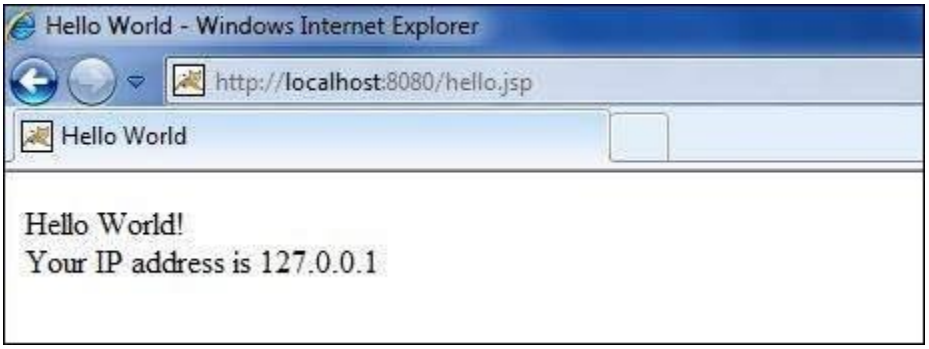
Any text, HTML tags, or JSP elements you write must be outside the scriptlet. Following is the simple and first example for JSP −

```
<html>
  <head><title>Hello World</title></head>

  <body>
    Hello World!<br/>
    <%
      out.println("Your IP address is " + request.getRemoteAddr());
    %>
  </body>
</html>
```

**NOTE** − Assuming that Apache Tomcat is installed in C:\apache-tomcat-7.0.2 and your environment is setup as per environment setup tutorial.

Let us keep the above code in JSP file **hello.jsp** and put this file in **C:\apache-tomcat7.0.2\webapps\ROOT** directory. Browse through the same using URL **http://localhost:8080/hello.jsp**. The above code will generate the following result −




## 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; ]+ ... %>
```

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

```
<jsp:declaration>
  code fragment
</jsp: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.

Because the value of an expression is converted to a String, you can use an expression within a line of text, whether or not it is tagged with HTML, in a 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 %>
```

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

```
<jsp:expression>
  expression
</jsp: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>
```

The above code will generate the following result −

Today's date: 11-Sep-2010 21:24:25

## 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>
```

The above code will generate the following result −

# A Test of Comments

There are a small number of special constructs you can use in various cases to insert comments or characters that would otherwise be treated specially. Here's a summary −

| S.No. | Syntax & Purpose |
|---|---|
| 1 | **<%-- comment --%>**<br>A JSP comment. Ignored by the JSP engine. |
| 2 | **<!-- comment -->**<br>An HTML comment. Ignored by the browser. |
| 3 | **<\%**<br>Represents static <% literal. |
| 4 | **%\>**<br>Represents static %> literal. |
| 5 | **\'**<br>A single quote in an attribute that uses single quotes. |
| 6 | **\"**<br>A double quote in an attribute that uses double quotes. |

## JSP Directives

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

<%@ directive attribute="value" %>

There are three types of directive tag −

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

We would explain the JSP directive in a separate chapter JSP - Directives

## 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" />
```

Action elements are basically predefined functions. Following table lists out the available JSP Actions −

| S.No. | Syntax & Purpose |
|---|---|
| 1 | **jsp:include**<br>Includes a file at the time the page is requested. |
| 2 | **jsp:useBean**<br>Finds or instantiates a JavaBean. |
| 3 | **jsp:setProperty**<br>Sets the property of a JavaBean. |
| 4 | **jsp:getProperty**<br>Inserts the property of a JavaBean into the output. |
| 5 | **jsp:forward**<br>Forwards the requester to a new page. |
| 6 | **jsp:plugin**<br>Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin. |
| 7 | **jsp:element**<br>Defines XML elements dynamically. |
| 8 | **jsp:attribute**<br>Defines dynamically-defined XML element's attribute. |
| 9 | **jsp:body**<br>Defines dynamically-defined XML element's body. |
| 10 | **jsp:text**<br>Used to write template text in JSP pages and documents. |

We would explain JSP actions in a separate chapter JSP - Actions

## JSP Implicit Objects

JSP supports nine automatically defined variables, which are also called implicit objects. These variables are −

| S.No. | Object & Description |
|---|---|
| 1 | **request**<br>This is the **HttpServletRequest** object associated with the request. |
| 2 | **response**<br>This is the **HttpServletResponse** object associated with the response to the client. |
| 3 | **out**<br>This is the **PrintWriter** object used to send output to the client. |
| 4 | **session**<br>This is the **HttpSession** object associated with the request. |
| 5 | **application**<br>This is the **ServletContext** object associated with the application context. |
| 6 | **config**<br>This is the **ServletConfig** object associated with the page. |
| 7 | **pageContext**<br>This encapsulates use of server-specific features like higher performance **JspWriters**. |
| 8 | **page**<br>This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class. |
| 9 | **Exception**<br>The **Exception** object allows the exception data to be accessed by designated JSP. |

We would explain JSP Implicit Objects in a separate chapter JSP - Implicit Objects.

## Control-Flow Statements

You can use all the APIs and building blocks of Java in your JSP programming including decision-making statements, loops, etc.

## Decision-Making Statements

The **if...else** block starts out like an ordinary Scriptlet, but the Scriptlet is closed at each line with HTML text included between the Scriptlet tags.

```
<%! int day = 3; %>
<html>
  <head><title>IF...ELSE Example</title></head>

  <body>
    <% if (day == 1 || day == 7) { %>
      <p> Today is weekend</p>
    <% } else { %>
      <p> Today is not weekend</p>
    <% } %>
  </body>
</html>
```

The above code will generate the following result −

Today is not weekend

Now look at the following **switch...case** block which has been written a bit differentlty using **out.println()** and inside Scriptletas −

```
<%! int day = 3; %>
<html>
  <head><title>SWITCH...CASE Example</title></head>

  <body>
    <%
      switch(day) {
        case 0:
          out.println("It\'s Sunday.");
          break;
        case 1:
          out.println("It\'s Monday.");
          break;
        case 2:
          out.println("It\'s Tuesday.");
          break;
        case 3:
          out.println("It\'s Wednesday.");
          break;
        case 4:
          out.println("It\'s Thursday.");
          break;
        case 5:
          out.println("It\'s Friday.");
          break;
        default:
          out.println("It's Saturday.");
      }
    %>
  </body>
</html>
```

The above code will generate the following result −

It's Wednesday.

## Loop Statements

You can also use three basic types of looping blocks in Java: **for, while, and do…while** blocks in your JSP programming.

Let us look at the following **for** loop example −

```
<%! int fontSize; %>
<html>
  <head><title>FOR LOOP Example</title></head>
```

```
  <body>
    <%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
      <font color = "green" size = "<%= fontSize %>">
        JSP Copyright(C) AlphaPeeler
    </font><br />
    <%}%>
  </body>
</html>
```

The above code will generate the following result −

JSP Copyright(C) AlphaPeeler
JSP Copyright(C) AlphaPeeler
JSP Copyright(C) AlphaPeeler

Above example can be written using the **while** loop as follows −

```
<%! int fontSize; %>
<html>
  <head><title>WHILE LOOP Example</title></head>

  <body>
    <%while ( fontSize <= 3){ %>
      <font color = "green" size = "<%= fontSize %>">
        JSP Copyright(C) AlphaPeeler
      </font><br />
      <%fontSize++;%>
    <%}%>
  </body>
</html>
```

The above code will generate the following result (output is wrong, please refer class code) −

JSP Copyright(C) AlphaPeeler
JSP Copyright(C) AlphaPeeler
JSP Copyright(C) AlphaPeeler

## JSP Operators

JSP supports all the logical and arithmetic operators supported by Java. Following table lists out all the operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom.

Within an expression, higher precedence operators will be evaluated first.

| Category | Operator | Associativity |
|---|---|---|
| Postfix | () [] . (dot operator) | Left to right |
| Unary | ++ - - ! ~ | Right to left |
| Multiplicative | * / % | Left to right |
| Additive | + - | Left to right |
| Shift | >> >>> << | Left to right |
| Relational | > >= < <= | Left to right |
| Equality | == != | Left to right |
| Bitwise AND | & | Left to right |
| Bitwise XOR | ^ | Left to right |
| Bitwise OR | \| | Left to right |

| | | |
|---|---|---|
| Logical AND | && | Left to right |
| Logical OR | \|\| | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= *= /= %= >>= <<= &= ^= \|= | Right to left |
| Comma | , | Left to right |

## 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