# JSP - Syntax



Previous Page Next Page

This tutorial will give basic idea on simple syntax (ie. elements) involved with JSP development:

## 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 XML equivalent of the above syntax as follows:

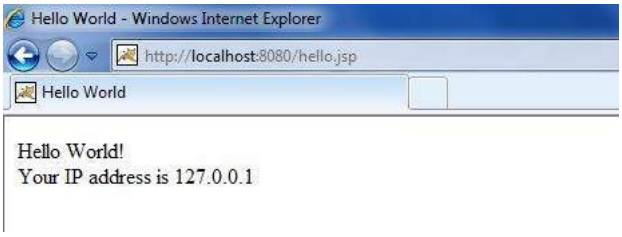
```
<jsp:scriptlet>
   code fragment
</jsp:scriptlet>
```

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 above code in JSP file hello.jsp and put this file in **C:\apache-tomcat-7.0.2\webapps\ROOT** directory and try to browse it by giving URL http://localhost:8080/hello.jsp. This would generate 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 of JSP Declarations:

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

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

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

Following is the simple 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 XML equivalent of the above syntax as follows:

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

Following is the simple example for JSP Expression:

```
<html>
<head><title>A Comment Test</title></head>
<body>
<p>
   Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
</body>
</html>
```

This would generate 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" part of your JSP page.

Following is the syntax of JSP comments:

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

Following is the simple example for 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>
```

This would generate 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:

| Syntax | Purpose |
|---|---|
| <%-- comment --%> | A JSP comment. Ignored by the JSP engine. |

| | |
|---|---|
| <!-- comment --> | An HTML comment. Ignored by the browser. |
| <\% | Represents static <% literal. |
| %\> | Represents static %> literal. |
| \' | A single quote in an attribute that uses single quotes. |
| \" | 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:

| Directive | Description |
|---|---|
| <%@ page ... %> | Defines page-dependent attributes, such as scripting language, error page, and buffering requirements. |
| <%@ include ... %> | Includes a file during the translation phase. |
| <%@ taglib ... %> | Declares a tag library, containing custom actions, used in the page |

We would explain JSP directive in 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 and there are following JSP actions available:

| Syntax | Purpose |
|---|---|
| jsp:include | Includes a file at the time the page is requested |
| jsp:useBean | Finds or instantiates a JavaBean |
| jsp:setProperty | Sets the property of a JavaBean |
| jsp:getProperty | Inserts the property of a JavaBean into the output |
| jsp:forward | Forwards the requester to a new page |
| jsp:plugin | Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin |
| jsp:element | Defines XML elements dynamically. |
| jsp:attribute | Defines dynamically defined XML element's attribute. |
| jsp:body | Defines dynamically defined XML element's body. |
| jsp:text | Use to write template text in JSP pages and documents. |

We would explain JSP actions in separate chapter JSP - Actions

## JSP Implicit Objects:

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

| Objects | Description |
|---|---|
| request | This is the **HttpServletRequest** object associated with the request. |

| | |
|---|---|
| response | This is the **HttpServletResponse** object associated with the response to the client. |
| out | This is the **PrintWriter** object used to send output to the client. |
| session | This is the **HttpSession** object associated with the request. |
| application | This is the **ServletContext** object associated with application context. |
| config | This is the **ServletConfig** object associated with the page. |
| pageContext | This encapsulates use of server-specific features like higher performance **JspWriters**. |
| page | This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class. |
| Exception | The **Exception** object allows the exception data to be accessed by designated JSP. |

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

## Control-Flow Statements:

JSP provides full power of Java to be embedded in your web application. 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 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>
```

This would produce 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>
```

This would produce 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 Tutorial
   </font><br />
<%}%>
</body>
</html>
```

This would produce following result:

JSP Tutorial

JSP Tutorial

JSP Tutorial

Above example can be written using **while** loop as follows:

```
<%! int fontSize; %>
<html>
<head><title>WHILE LOOP Example</title></head>
<body>
<%while ( fontSize <= 3){ %>
   <font color="green" size="<%= fontSize %>">
    JSP Tutorial
   </font><br />
<%fontSize++;%>
<%}%>
</body>
</html>
```

This would also produce following result:

JSP Tutorial

JSP Tutorial

JSP Tutorial

## JSP Operators:

JSP supports all the logical and arithmetic operators supported by Java. Following table give a list of 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

Previous Page Next Page