The Scriptlet:

<% code fragment %>

<jsp:scriptlet>

code fragment

</jsp:scriptlet>

## JSP Declarations:

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

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

<jsp:declaration>

code fragment

</jsp:declaration>

## 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>

## JSP Comments:

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

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. Attributes eg: buffer, contenttype, import, extends, language, isthreadsafe, session |
| <%@ include ... %> | Includes 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. |
| <%@ taglib ... %> | Declares a tag library, containing custom actions, used in the page  <%@ taglib uri="uri" prefix="prefixOfTag" > |

JSP Actions: I usg fp eabt

I use Fb while eating.

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. |

JSP Implicit Objects:

Rrosacppe (Rosacpe)

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**. This object is intended as a means to access information about the page while avoiding most of the implementation details. The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope. |
| 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. |

**GET POST - answer**

**First 2 para in below url - important**

[**http://www.tutorialspoint.com/jsp/jsp\_form\_processing.htm**](http://www.tutorialspoint.com/jsp/jsp_form_processing.htm)

# JSP – Filters

Servlet and JSP Filters are Java classes that can be used in Servlet and JSP Programming for the following purposes:

* To intercept requests from a client before they access a resource at back end.
* To manipulate responses from server before they are sent back to the client.

There are various types of filters suggested by the specifications:

* Authentication Filters.
* Data compression Filters
* Encryption Filters .
* Filters that trigger resource access events.
* Image Conversion Filters .

**COOKIES:**

Cookies are text files stored on the client computer and they are kept for various information tracking purpose. JSP transparently supports HTTP cookies using underlying servlet technology.

There are three steps involved in identifying returning users:

* Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
* Browser stores this information on local machine for future use.
* When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user or may be for some other purpose as well.

A JSP that sets a cookie might send headers that look something like this:

HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;

path=/; domain=tutorialspoint.com

Connection: close

Content-Type: text/html

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server.

# JSP - Standard Tag Library (JSTL) Tutorial

[**http://www.tutorialspoint.com/jsp/jsp\_standard\_tag\_library.htm**](http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm)

* **Core Tags –**most frequently used JSTL tags
  + if, when, otherwise, choose, set ,out, param , url, redirect, foreach , fortokens, import, catch, remove
* **Formatting tags-** used to format and display text
  + parseNumber, formatNumber,parseDate, formatDate, message, bundle, setBundle, setLocale, timeZone, setTimeZone, requestEncoding
* **SQL tags –**  tags for interacting with relational databases
  + query, update, param, dateParam, setDataSource, transaction
* **XML tags –** JSP-centric way of creating and manipulating XML documents
  + if, when , otherwise, choose, set, out, param, parse, foreach
* **JSTL Functions -** string manipulation functions
  + contains, containsIgnoreCase, split, trim, replace, endsWith, startsWith, length, substring, subStringAfter, subStringBefore, indexOf, join, toUpperCase, toLowerCase.

**Custom Tags:**

To write a custom tab you can simply extend SimpleTagSupport class and override the **doTag()** method, where you can place your code to generate content for the tag. To create a custom JSP tag, you must first create a Java class that acts as a tag handler.

[**http://www.tutorialspoint.com/jsp/jsp\_custom\_tags.htm**](http://www.tutorialspoint.com/jsp/jsp_custom_tags.htm)