**JavaServer Pages**

JavaServer Pages is a server-side programming that allows use of Java code to create dynamic web pages that are platform independent. JSP is able to use existing Java APIs like the jdbc for accessing databases.

JSP can be used in many ways such as collecting user input from a form, retrieving and updating databases, creating dynamic content and passing control through forward or inclusion mechanism.

JSP is an integral part of Java EE, a complete platform for enterprise application, which makes it very flexible in the sense that it can be used to create applications that are simple and also can be used for very complex requirements.

**How to get started with JSP**

1. Install a web server that can support JSP and Servlet, there are some that are available in the market and that are free like Apache Tomcat.
2. We recommend you to use Apache Tomcat for the moment, so start setting it up by downloading it from <https://tomcat.apache.org/>
3. Once done downloading, unpack the binary distribution and create a new environment variable in your system named CATALINA\_HOME with the directory of your tomcat.
4. Start apache tomcat by typing the commands %CATALINA\_HOME%\bin\startup.bat or (root directory of where your tomcat is located) example C:\apache-tomcat-5.5.29\bin\startup.bat
5. After successfully starting tomcat, you can now check it through your browser by accessing <http://localhost:8080/>

* If successful you will be seeing some information about Apache Software Foundation.
* There are platforms that are supports Apache Tomcat such as NetBeans and Eclipse.

1. Setup a classpath by editing autoexec.bat and adding the following code:

set CATALINA = C:\apache-tomcat-5.5.29

set CLASSPATH = %CATALINA%\common\lib\jsp-api.jar;%CLASSPATH%

or by opening My Computer > Properties > Advanced > Environment Variables then updating the classpath value.

**Elements of JSP**

1. Scriplet – a piece of java code embedded in a html-like JSP code

Syntax:

<% arbitrary chunk of java code %>

Syntax when with HTML tags:

<html>

<head>

<title> Scriplet </title>

</head>

<body>

<%

out.println(“Scriplet inside html”);

%>

</body>

</html>

1. Declaration – use to declare variables and methods in a page’s scripting language

Syntax:

<%! Scripting-language-declaration %>

Examples:

<%! int x = 0; %>

<%! int var1, var2, var3; %>

1. Expression – used to insert value of scripting language expression, converted into string, into the data stream returned to the client.

Syntax: <%= expression %>

Syntax with HTML:

<html>

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

<body>

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

</body>

</html>

1. Comments – to put JSP comment that are ignored by JSP engine.

Syntax: <%-- This is a JSP comment --%>

1. Directives - instructions that tell the container how to translate JSP to servlet code

Syntax: <%@ directive attribute=”value” %>

Types:

1. Page – setting for the entire jsp page

Syntax: <%@ page %>

Some Examples:

1. Import attribute - most commonly used page directive attribute for importing other java classes, interfaces and others.

<%@ page import=”java.util.Date” %>

1. contentType attribute – use to set content type and character set of the response message

<%@ page contentType="text/html; charset=US-ASCII" %>

1. pageEncoding attribute – use to set encoding type of the response

<%@ page pageEncoding="US-ASCII" %>

1. language attribute – use to specify the scripting language of the JSP page

<%@ page language="java" %>

1. extends attribute – used to specify the base class of a derived servlet code.

<%@ page extends="org.apache.jasper.runtime.HttpJspBase" %>

1. include – used to integrate contents of another file to the current JSP page

<%@ include file="test.html" %>

1. taglib – used to define a tag library

<%@ taglib uri="/WEB-INF/c.tld" prefix="c"%>

1. Actions – used to control flow between pages and to use Java Bean

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

Examples:

1. jsp:include – includes another file
2. jsp:forward – forward request/response to another page
3. jsp:setProperty – sets the value of a JavaBean property
4. jsp:getProperty – retrieves and prints the value of a JavaBean property
5. jsp:useBean – creates or finds a bean object
6. jsp:plugin – embed another component
7. jsp:param – used to set parameter value
8. Implicit Objects – created by JSP engine and are automatically defined
9. request – **HttpServletRequest** object associate with the request
10. response – **HttpServletResponse** object associated with the response
11. out – **PrintWriter** object used to print output to the client
12. session – **HttpSession** object associated with the request
13. application – **ServletContext** object associated with the application context
14. config – **ServletConfig** object associated with the page
15. pageContext – **PageContext** object that encapsulates server-specific features
16. page – **Object** similar to **this** keyword
17. exception – **Exception** object that allows the exception data to be accessed by selected JSP.

JSP Client Request

1. The HTTPServletRequest Object

Request object that is an instance of javax.servlet.http.HttpServletRequest object. This is created by the JSP Engine once there is a request.

Some methods that can be used to read HTTP header

1. HttpSession getSession() – returns the current session associated with the request, if there is no session existing then a session will be created
2. Cookie[] getCookies()

**Request Dispatching**

The problem with how some web pages implement request dispatching is that codes are not reusable (god class) and too many work are done by one servlet. Also, codes will be repeated through each pages and if there will be some changes, for example in the navigation bar, then every pages will have to be updated so it’s difficult to maintain.

Solution is the Divide and Conquer method

Type:

1. Forward Mechanism – servlets are given roles/work to do and each output of servlets will be given to another servlet. Control is pass through and lost by a servlet and is given to another servlet.
2. Include Mechanism – different servlets have their own role/work to do and one servlet will serve as the combinator of every output of the other servlets. Control is maintained in each servlet.

References:

<http://docs.oracle.com/javaee/5/tutorial/doc/bnaos.html>

<http://www.oracle.com/technetwork/java/javaee/jsp/index.html>

<http://www.journaldev.com/2044/jsp-directives-page-include-taglib-example>

<https://www.tutorialspoint.com/jsp/>

https://www.javatpoint.com/jsp-action-tags-forward-action