**JAVA SERVER PAGES(JSP)**

JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>

A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

**Advantage of JSP over Servlet:**

1) Extension to Servlet

2) Easy to maintain

3) Fast Development: No need to recompile and redeploy

4) Less code than Servlet

**Life cycle of a JSP Page:**

1)Translation of JSP Page

2)Compilation of JSP Page

3)Classloading🡪 (class file is loaded by the classloader)

4)Instantiation 🡪(Object of the Generated Servlet is created).

5)Initialization 🡪( jspInit() method is invoked by the container).

6)Reqeust processing🡪 ( \_jspService() method is invoked by the container).

7)Destroy 🡪( jspDestroy() method is invoked by the container).

**index.jsp**

<html>

<body>

<% out.print(2\*5); %>

</body>

</html>

=====================================================================================

**the JSP API consists of two packages:**

1)javax.servlet.jsp

2)javax.servlet.jsp.tagext

The javax.servlet.jsp package has two interfaces and classes.The two interfaces are as follows:

**JspPage**

**HttpJspPage**

The classes are as follows:

1)JspWriter

2)PageContext

3)JspFactory

4)JspEngineInfo

5)JspException

6)JspError

7)JSP Scriptlet tag

8)JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

**1)scriptlet tag**

**2)expression tag**

**3)declaration tag**

**1)JSP scriptlet tag**

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

<%  java source code %>

**Example of JSP scriptlet tag**

<html>

<body>

<% out.print("welcome to jsp"); %>

</body>

</html> #

=====================================================================================

**JSP expression tag**

The code placed within JSP expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

**Example of JSP expression tag**

<html>

<body>

<%= "welcome to jsp" %>

</body>

</html> #

=====================================================================================

<html>

<body>

Current Time: <%= java.util.Calendar.getInstance().getTime() %>

</body>

</html> #

======================================================================================

**JSP Declaration Tag**

The JSP declaration tag is used to declare fields and methods.The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.So it doesn't get memory at each request.

**index.jsp**

<html>

<body>

<% int data=50; %>

<%= "Value of the variable is:"+data %>

</body>

</html> #

=====================================================================================

**implicit objects**

There are 9 jsp implicit objects. These objects are created by the web container that are available to all the jsp pages.

|  |  |
| --- | --- |
| **Object** | **Type** |
| out | JspWriter |
| request | HttpServletRequest |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| page | Object |
| exception | Throwable |

**1) JSP out implicit object**

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write

**index.jsp**

<html>

<body>

<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>

</body>

</html>

========================================================================================

**2)JSP request implicit object**

The JSP request is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.It can also be used to set, get and remove attributes from the jsp request scope.

**Example of JSP request implicit object:**

**index.html**

<form action="welcome.jsp">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

**welcome.jsp**

<%

String name=request.getParameter("uname");

out.print("welcome "+name);

%>

========================================================================================

**3) JSP response implicit object**

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.It can be used to add or manipulate response such as redirect response to another resource, send error etc.

**Example of response implicit object**

**index.html**

<form action="welcome.jsp">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

**welcome.jsp**

<%

response.sendRedirect("http://www.google.com");

%>

=======================================================================================

**4) JSP config implicit object**

In JSP, config is an implicit object of type ServletConfig. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

**Example of config implicit object:**

index.html

<form action="welcome">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

web.xml file

<web-app>

<servlet>

<servlet-name>alok</servlet-name>

<jsp-file>/welcome.jsp</jsp-file>

<init-param>

<param-name>dname</param-name>

<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>alok</servlet-name>

<url-pattern>/welcome</url-pattern>

</servlet-mapping>

</web-app>

**welcome.jsp**

<%

out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");

out.print("driver name is="+driver);

%>

========================================================================================

**5) JSP application implicit object**

In JSP, application is an implicit object of type ServletContext.

The instance of ServletContext is created only once by the web container when application or project is deployed on the server.

This object can be used to get initialization parameter from configuaration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

**Example of application implicit object:**

**index.html**

<form action="welcome">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

**web.xml file**

<web-app>

<servlet>

<servlet-name>alok</servlet-name>

<jsp-file>/welcome.jsp</jsp-file>

</servlet>

<servlet-mapping>

<servlet-name>alok</servlet-name>

<url-pattern>/welcome</url-pattern>

</servlet-mapping>

<context-param>

<param-name>dname</param-name>

<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>

</context-param>

</web-app>

**welcome.jsp**

<%

out.print("Welcome "+request.getParameter("uname"));

String driver=application.getInitParameter("dname");

out.print("driver name is="+driver);

%>

========================================================================================

**6) session implicit object**

In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information.

Example of session implicit object

**index.html**

<html>

<body>

<form action="welcome.jsp">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

</body>

</html>

**welcome.jsp**

<html>

<body>

<%

String name=request.getParameter("uname");

out.print("Welcome "+name);

session.setAttribute("user",name);

<a href="second.jsp">second jsp page</a>

%>

</body>

</html>

**second.jsp**

<html>

<body>

<%

String name=(String)session.getAttribute("user");

out.print("Hello "+name);

%>

</body>

</html>

========================================================================================

**7) pageContext implicit object**

In JSP, pageContext is an implicit object of type PageContext class.The pageContext object can be used to set,get or remove attribute from one of the following scopes:

page

request

session

application

In JSP, page scope is the default scope.

Example of pageContext implicit object

**index.html**

<html>

<body>

<form action="welcome.jsp">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

</body>

</html>

**welcome.jsp**

<html>

<body>

<%

String name=request.getParameter("uname");

out.print("Welcome "+name);

pageContext.setAttribute("user",name,PageContext.SESSION\_SCOPE);

<a href="second.jsp">second jsp page</a>

%>

</body>

</html>

**second.jsp**

<html>

<body>

<%

String name=(String)pageContext.getAttribute("user",PageContext.SESSION\_SCOPE);

out.print("Hello "+name);

%>

</body>

</html>

=================================================================================

**jsp directives**

The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

**1)page directive**

**2)include directive**

**3)taglib directive**

=====================================================================================

**JSP page directive**

The page directive defines attributes that apply to an entire JSP page.

**<%@ page attribute="value" %>**

**Attributes of JSP page directive**

import

contentType

extends

info

buffer

language

isELIgnored

isThreadSafe

autoFlush

session

pageEncoding

errorPage

isErrorPage

**1)import**

The import attribute is used to import class,interface or all the members of a package.It is similar to import keyword in java class or interface.

<html>

<body>

<%@ page import="java.util.Date" %>

Today is: <%= new Date() %>

</body>

</html>

=======================================================================================

**2)contentType**

The contentType attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.The default value is "text/html;charset=ISO-8859-1".

**Example of contentType attribute**

<html>

<body>

<%@ page contentType=application/msword %>

Today is: <%= new java.util.Date() %>

</body>

</html>

**3)extends**

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

**4)info**

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

**Example of info attribute**

<html>

<body>

<%@ page info="composed by Alok Bhatt" %>

Today is: <%= new java.util.Date() %>

</body>

</html>

The web container will create a method getServletInfo() in the resulting servlet.For example:

public String getServletInfo() {

return "composed by alok";

}

**5)buffer**

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

Example of buffer attribute

<html>

<body>

<%@ page buffer="16kb" %>

Today is: <%= new java.util.Date() %>

</body>

</html>

**6)language**

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

**7)isELIgnored**

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. Expression Language is enabled by default. We see Expression Language later.

<%@ page isELIgnored="true" %>//Now EL will be ignored

**8)isThreadSafe**

Servlet and JSP both are multithreaded.If we want to control this behaviour of JSP page, we can use **isThreadSafe** attribute of page directive.The value of isThreadSafe value is true.If we make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.If you make the value of isThreadSafe attribute like:

**<%@ page isThreadSafe="false" %>**

The web container in such a case, will generate the servlet as:

public class SimplePage\_jsp extends HttpJspBase

implements SingleThreadModel{

.......

}

**9)errorPage**

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

**Example of errorPage attribute**

**//index.jsp**

<html>

<body>

<%@ page errorPage="myerrorpage.jsp" %>

<%= 100/0 %>

</body>

</html>

**10)isErrorPage**

The isErrorPage attribute is used to declare that the current page is the error page.

Note: The exception object can only be used in the error page.

**Example of isErrorPage attribute**

**//myerrorpage.jsp**

<html>

<body>

<%@ page isErrorPage="true" %>

Sorry an exception occured!<br/>

The exception is: <%= exception %>

</body>

</html>

========================================================================================

**Jsp Include Directive**

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource)

<%@ include file="resourceName" %>

**Example of include directive**

<html>

<body>

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

Today is: <%= java.util.Calendar.getInstance().getTime() %>

</body>

</html>

=======================================================================================

**taglib directive JSP**

The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

**Example of JSP Taglib directive**

In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

<html>

<body>

<%@ taglib uri="http://www.google.com/tags" prefix="mytag" %>

<mytag:currentDate/>

</body>

</html>

========================================================================================

**JSP Action Tags**

There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks.

**JSP Action Tags Description**

**jsp:forward** forwards the request and response to another resource.

jsp:include includes another resource.

jsp:useBean creates or locates bean object.

jsp:setProperty sets the value of property in bean object.

jsp:getProperty prints the value of property of the bean.

jsp:plugin embeds another components such as applet.

jsp:param sets the parameter value. It is used in forward and include mostly.

jsp:fallback can be used to print the message if plugin is working. It is used in jsp:plugin.

========================================================================================

**Example of jsp:forward action tag without parameter**

In this example, we are simply forwarding the request to the printdate.jsp file.

**index.jsp**

<html>

<body>

<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" />

</body>

</html>

================================================================================

**printdate.jsp**

<html>

<body>

<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>

</body>

</html>

=====================================================================================

**Example of jsp:forward action tag with parameter**

**index.jsp**

<html>

<body>

<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" >

<jsp:param name="name" value="ab" />

</jsp:forward>

</body>

</html>

**printdate.jsp**

<html>

<body>

<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>

<%= request.getParameter("name") %>

</body>

</html>

========================================================================================

**jsp:include action tag**

The jsp:include action tag is used to include the content of another resource it may be jsp, html or servlet.The jsp include action tag includes the resource at request time so it is better for dynamic pages because there might be changes in future.

**The jsp:include tag can be used to include static as well as dynamic pages.**

**Index.jsp**

<h2>this is index page</h2>

<jsp:include page="printdate.jsp" />

<h2>end section of index page</h2>

=====================================================================================

**jsp:useBean action tag**

The jsp:useBean action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.