**Introduction to jsp(java server page)**

* jsp is a server side technology used to design dynamic web pages.
* These dynamic webpages can be created either by Servlets or jsp.
* Jsp is not only alternative to Servlets but also has benefits over Servlets.
* Every jsp file has to be saved with .jsp extension.

**Advantages with jsp over Servlets:**

* Jsp is a html program which contains java code so no need to compile the jsp program.
* If any changes are made to the jsp program no need to recompile the jsp program.
* No need to redeploy & reload the application if any changes are made.
* Jsp is more flexible compared to the Servlet.

**Location of jsp file in web application:**

webapplication

C classes

lib

src

Web\_inf

.html

Mypage1.jsp

Mypage2.jsp

Web.xml

**Execution flow of jsp program:**

Browser sends request for jsp file

Response will be send back to the browser

Executes service() method

Executes init()method

Request,response objects will be created

Object will be created for converted Servlet by web container

Jsp engine will convert jsp file into Servlet program

**Jsp api**

* Following are the pre-defined classes and interfaces present in javax.servlet.jsp package to work with jsp files.

|  |  |
| --- | --- |
| **Interfaces**  1.JspPage  2.HttpJspPage | **Classes**  **1**.JspWriter  2.JspContext  3.PageContext  4.JspFactory  5.JspEngineInfo  6.ErrorData |

* The above classes and interfaces are never used in writing jsp pages.
* But these classes and interfaces are implicitly used by the webcontainer.
* To design web application using jsp,isp provides implicit objects and these objects are directly used while writing a jsp file.

Following are the list of implicit objects provided by the jsp api

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | | | **Object name** |
| JspWriter | | | Out |
| HttpServletRequest | | | Request |
| HttpServletResponse | | Response |
| HttpSession | | session |
|  | Throwable exception | |

1.out

It is an predefined object for JspWriter class used to display any content on web browser

Ex:- out.print(“”);out.println(“”);

2.request

It is an implicit object for HttpServletRequest class used to handle the request given by the client.

Ex: request.getParameter();

3.response

It is an implicit object of HttpServletResponse used to handle the response data

Ex: response.sendRedirect();

4.session

It is an implicit object of HttpSession used to work with sessionTracking

Ex: session.setAttribute()//used to set value to session object

5.exception

It is an implicit object of throwable class used to display the exception messages which was raised in jsp page.

Ex: out.println(exception);

**Jsp elements (or) tags:**

To write the java code in jsp following pre-defined tags are used.

**1.Scripting elements**

**2.directive elements**

**1.Scripting elements**

These elements provides the ability to insert the java code int a jsp file,these are classified into the following types

1. Expression tag
2. Declarative tag
3. Scriplet tag
4. Comment tag

**Expression tag**

It is used to display the messages on webpage

Syntax:

<%=”msg”%>

Now the above statement is converted with the following Servlet code

PrintWriter pw=response.getWriter();

Pw.println(msg);

Design a web application to display the welcome message using jsp

Index.html:

<html>

<body>

<form name=”fm” action=”welcome.jsp”>

Name:<input type=”text” name=”nm”>

<input type=”submit” value=”submit”>

</form>

</body>

</html>

Welcome.jsp

<html>

<body>

<%=request.getParameter(“nm”)%>

</body>

</html>

ii)Declarative tag

it is used to write the declaration code in jsp like variable declaration,method declaration etc..

syntax

<%! Datatype variable name;

F1()

{

}

%>

If the code was written inside the declarative element then that will be placed outside the service method, but inside the converted class of Servlet.

Design a web application using jsp which displays the square of a number

<html>

<body>

<%! int v=5,r;

Void square()

{

r=v\*v;

}

%>

<%= out.println(r) %>

iii.Scriplet tag

it is used to write the business logic in jsp program .the code written in scriplet tag will converted to service method in the converted Servlet.

Syntax:

<% business logic %>

Design a web application to insert a record in database table using jsp

Index.html

<html>

<body>

<form name=”fm” action=”welcome.jsp”>

Name:<input type=”text” name=”nm”>

rollno:<input type=”text” name=”rn”>

<input type=”submit” value=”submit”>

</form>

</body>

</html>

Welcome.jsp

<html>

<body>

<[%@page import=”java.sql.\*”%](mailto:%25@page%20import=)>

<%

Try

{

String name=request.getParameter(“nm”);

String r=request.getParameter(“rn”);

Int rollno=Integer.parseInt(r);

Class.forName(“oracle.jdbc.driver.OracleDriver”);

Connection con=DriverManager.getConnection(“jdbc:oracle:thin:@localhost:1521:XE”,”scott”,”tiger”);

Statement stmt=con.createStatement();

int i=stmt.executeUpdate(“insert into student values(“+name+”,”+rollno+”)”);

if(i>0)

<%= out.println(“records inserted successfully…”);%>

Else

<%= out.println(“records not inserted…”);%>

}

Catch(Exception e)

{

<%= out.println(e);%>

}

%>

</body>

</html>

iv) comment tag

JSPs provides two types of comments for putting comment in your page.

(1)First type of comment is for output comment which is appeared in the output stream on the browser. It is written by using the “<!--comments-->” tags.

Syntax

<!--comments-->

(2)Second type of comment is not delivered to the browser. It is written by using the “<%-- comments --%>” tags.

Syntax –

<%-- comments --%>

**2.Directive Elements:**

These are the elements supported by the jsp which tells the web container how the jsp has to be translated into Servlet.

These directive elements are classified into following types:

i.include directive

ii.page directive

i.include directive:

it is used to include either html or jsp file within our jsp file.

Syntax:

<%@ include file=”resourcename”%>

This resource name can be a html or jsp file.

Design a web application to include header.jsp,footer.jsp into index.jsp

Header.jsp

<html>

<body>

<h1>

This is header of jsp

</h1></body></html>

Footer.jsp

<html>

<body>

<h1>

This is footer of jsp

</h1></body>

</html>

Index.jsp

<html>

<body>

<%@ include file=”header.jsp”%>

<h2>welcome to jsp..</h2>

<%@ include file=”footer.jsp”%>

</body>

</html>

ii. page directive

These are the special elements (or) tags with set of attributes that applies to the entire jsp page

Syntax

<%@ page attribute=”value”%>

In the above syntax attribute can be any of the following:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **description** | **syntax** |
| Import | Used to import packages in the current jsp file | <%@ page import=”javax.sql.\*”%> |
| Content type | Used to set the response of jsp page | <%@ page content type=”text/html”%> |
| Session | Used to enable or disable the http session for session handling | <%@ page session=”true”%>  True-enable,false-disable |
| Errorpage | Used to define the error jsp that means if any exception is raised in the jsp file that will be redirected to given error jsp file | <%@ page errorpage=”errorfile.jsp”%> |
| isErrorPage | It is used to set a jsp page as an error page | <%@ page isErrorPage=”true”%>  True-error jsp page  False-normal jsp page |

**JSTL (java server pages standard tag library):**

* It is used to design the jsp pages by eliminating declaration tags and scriplet tags.
* Jstl tags are predefined and almost seems like a html tags.
* To work with jstl tags in our jsp page we need to download “jstl.jar” file and place in the “lib ” folder in our web application.
* To include jstl tags in our jsp at the top of the jsp we need to include <tagtlib> directive.
* Jstl tags are mainly categorized into the following types:

1. Jstl core tags
2. Jstl format tags
3. Jstl xml tags
4. Jstl function tags
5. Jstl sql tags

1.jstl core tags

These tags can be used to write the basic java code without using scriplet or declarative tags.

<%@ taglib prefix="c" uri = http://java.sun.com/jsp/jstl/core" %>

Following are the list of tags supported by the jstl core library.

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | **<c:out >**  Displays the output.  Like <%= ... >, expressions on web page. |
| 2 | **<c:set >**  Sets the result of an expression evaluation in a **'scope'** |
| 3 | **<c:remove >**  Removes a **scoped variable** (from a particular scope, if specified). |
| 4 | **<c:catch>**  Catches any **Throwable** that occurs in its body and optionally exposes it. |
| 5 | **<c:if>**  Simple conditional tag which evalutes its body if the supplied condition is true. |
| 6 | **<c:choose>**  It acts as switch statement & must be associated with**<when>** and **<otherwise>**. |
| 7 | **<c:when>** |
| Subtag of **<choose>** that includes its body if its condition evalutes to **'true'**. | |
| 8 | **<c:otherwise >**  Subtag of **<choose>** that follows the **<when>** tags and runs only if all of the prior conditions evaluated to **'false'**. |
| 9 | **<c:import>**  Retrieves an absolute or relative URL and exposes its contents to either the page, a String in **'var'**, or a Reader in **'varReader'**. |
| 10 | **<c:forEach >**  The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality . |
| 11 | **<c:forTokens>**  Iterates over tokens, separated by the supplied delimeters. |
| 12 | **<c:param>**  Adds a parameter to a containing **'import'** tag's URL. |
| 13 | **<c:redirect >**  Redirects to a new URL. |
| 14 | **<c:url>**  Creates a URL with optional query parameters |

**2.Formatting Tags**

The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Websites. Following is the syntax to include Formatting library in your JSP

<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>

Following table lists out the Formatting JSTL Tags

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | **<fmt:formatNumber>**  To render numerical value with specific precision or format. |
| 2 | **<fmt:parseNumber>**  Parses the string representation of a number, currency, or percentage. |
| 3 | **<fmt:formatDate>**  Formats a date and/or time using the supplied styles and pattern. |
| 4 | **<fmt:parseDate>**  Parses the string representation of a date and/or time |
| 5 | **<fmt:bundle>**  Loads a resource bundle to be used by its tag body. |
| 6 | **<fmt:setLocale>**  Stores the given locale in the locale configuration variable. |
| 7 | **<fmt:setBundle>**  Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable. |
| 8 | **<fmt:timeZone>** |
| Specifies the time zone for any time formatting or parsing actions nested in its body. | |
| 9 | **<fmt:setTimeZone>**  Stores the given time zone in the time zone configuration variable |
| 10 | **<fmt:message>**  Displays an internationalized message. |
| 11 | **<fmt:requestEncoding>**  Sets the request character encoding |

**3.SQL Tags**

The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as **Oracle, mySQL**, or **Microsoft SQLServer**.

Following is the syntax to include JSTL SQL library in your JSP

<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>

Following table lists out the SQL JSTL Tags

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | **<sql:setDataSource>**  Creates a simple DataSource suitable only for prototyping |
| 2 | **<sql:query>**  Executes the SQL query defined in its body or through the sql attribute. |
| 3 | **<sql:update>** |

Executes the SQL update defined in its body or through the sql attribute.

|  |  |
| --- | --- |
| 4 | **<sql:param>**  Sets a parameter in an SQL statement to the specified value. |
| 5 | **<sql:dateParam>**  Sets a parameter in an SQL statement to the specified java.util.Date value. |
| 6 | **<sql:transaction >**  Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction. |

**XML tags**

The JSTL XML tags provide a JSP-centric way of creating and manipulating the XML documents. Following is the syntax to include the JSTL XML library in your JSP.

The JSTL XML tag library has custom tags for interacting with the XML data. This includes parsing the XML, transforming the XML data, and the flow control based on the XPath expressions.

<%@ taglib prefix = "x" uri = "http://java.sun.com/jsp/jstl/xml" %>

Following is the list of XML JSTL Tags

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | **<x:out>**  Like <%= ... >, but for XPath expressions. |

|  |  |
| --- | --- |
| 2 | **<x:parse>**  Used to parse the XML data specified either via an attribute or in the tag body. |
| 3 | **<x:set >**  Sets a variable to the value of an XPath expression. |
| 4 | **<x:if >**  Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored. |
| 5 | **<x:forEach>**  To loop over nodes in an XML document. |
| 6 | **<x:choose>**  Simple conditional tag that establishes a context for mutually exclusive conditional operations,marked by **<when>** and **<otherwise>** tags. |
| 7 | **<x:when >**  Subtag of **<choose>** that includes its body if its expression evalutes to 'true'. |
| 8 | **<x:otherwise >**  Subtag of **<choose>** that follows the **<when>** tags and runs only if all of the prior conditions evaluates to 'false'. |
| 9 | **<x:transform >**  Applies an XSL transformation on a XML document |

**5.JSTL Functions**

JSTL includes a number of standard functions, most of which are common string manipulation functions.

Following is the syntax to include JSTL Functions library in your JSP

<%@ taglib prefix = "fn" uri = "http://java.sun.com/jsp/jstl/functions" %>

Following table lists out the various JSTL Functions

|  |  |
| --- | --- |
| **S.No.** | **Function & Description** |
| 1 | **fn:contains()**  Tests if an input string contains the specified substring. |
| 2 | **fn:containsIgnoreCase()**  Tests if an input string contains the specified substring in a case insensitive way. |
| 3 | **fn:endsWith()**  Tests if an input string ends with the specified suffix. |
| 4 | **fn:escapeXml()**  Escapes characters that can be interpreted as XML markup. |

|  |  |
| --- | --- |
| 5 | **fn:indexOf()**  Returns the index withing a string of the first occurrence of a specified substring. |
| 6 | **fn:join()**  Joins all elements of an array into a string. |
| 7 | **fn:length()**  Returns the number of items in a collection, or the number of characters in a string. |
| 8 | **fn:replace()**  Returns a string resulting from replacing in an input string all occurrences with a given string. |