**JSP (Java Server Pages)**

**JSP:(Java Server Page)**

**---------------------**

**-> Jsp is a standard java extension used to simplify the creation and management of dynamic web pages.**

**-> Jsps allow us to seperate the dynamic content of a Web page from its static presentation content.**

**-> As described earlier, programming in servlets is very complex and requires additional files, such as web.xml and java class files, to generate the required Web pages.**

**-> Jsp offers an easier approach to build dynamic web pages.**

**-> Jsp consist of HTML tags and special tags,known as JSP tags.**

**-> HTML tags are used to create static page content and JSP tags are used to add dynamic content to Web pages.**

**-> JSP pages are compiled into a java servlet by a JSP transaltor.**

**-> This java servlet is then compiled and executed to generate an output for the browser(client).**

**Advantages of Jsp:-**

**-------------------**

**1.Jsp's are easy to develop.We can eliminate the java code completely from the jsp's.**

**2.A developer has no experience in java can use the jsp technology by using jsp's.**

**3.As it is easy to develop the web application,the time required for developing web application using jsp's is lesser than the time required for developing the web application using manually developing servlets.**

**->We can seperate the presentation logic from the business logic easily by using jsp's.**

**Stages of Lifecycle of JSP are as follows:**

**------------------------------------------**

**a) Page Translation :**

**In this stage, The WC performs the following operations:**

**-> Locates the requested JSP page**

**-> Validates the syntactic correctness of the JSP Page.**

**-> Interprets the standard JSP directives,actions, and custom actions used in the JSP page.**

**-> Writes the source code of the equivalent Servlet for the JSP page.**

**b) Compilation:**

**In this stage, the JSP compiler compiles the java source code for the corresponding Servlet and converts it into java byte(class) code.**

**c) Loading & Initialization:**

**-> if the Loading and Initialization stage is performed successfully, the Web Container activates the servlet object and makes it available and ready to handle client requests.**

**d) Request Handling:**

**-> here \_jspService(request,response) will be executed.**

**e) Destroying( End of service ):**

**-> here \_jspDestroy() method will be executed.**

**JSP ELEMENTS:**

**-------------**

**->We can use various types of jsp elements in a jsp page.**

**The jsp elements are:-**

**1.Template text**

**2.Scriptlet (<%........%> )**

**3.Jsp expressions ( <%= .........%> )**

**4.jsp directives ( <%@ .........%> )**

**5.jsp declarations (<%! ............%> )**

**6.jsp Action tags**

**7.Jsp custom tags**

**8.EL(Expression language)expressions.(Added as part of jsp2.0)**

**sample jsp page:(test.jsp)**

**----------------**

**<html>**

**<head>**

**<title>This is first Jsp Page</title>**

**</head>**

**<body>**

**<p>Paragraph One</p>**

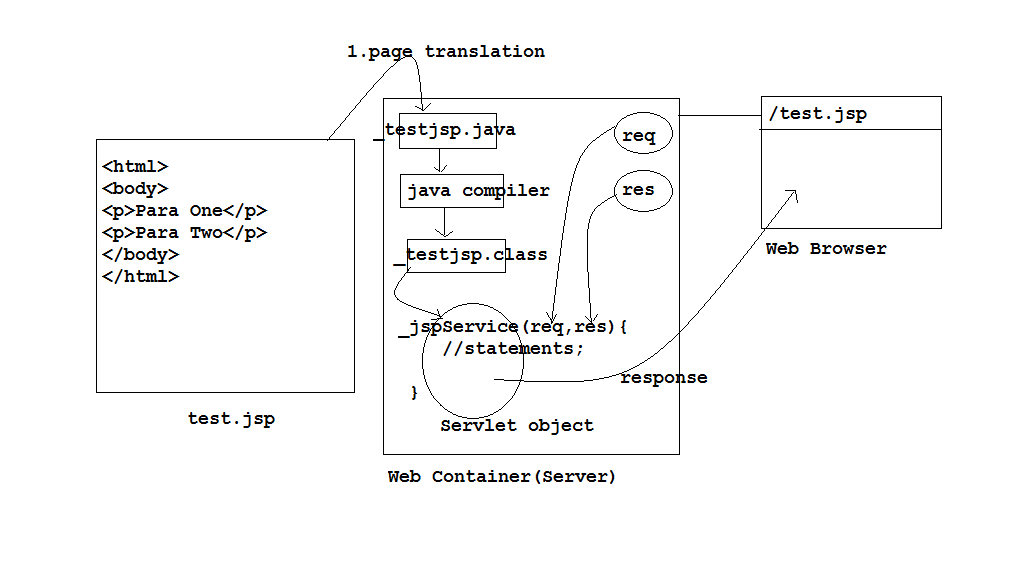
**<p>Paragraph Two</p>**

**</body>**

**</html>**

**JSP Request Flow Diagram: -**

**-------------------------------------**

****

**Using template text and scriptlet:**

**----------------------------------**

**code:-(one.jsp)**

**-----**

**This is first line of template text<br>**

**<%**

**System.out.println("This is scriptlet");**

**%>**

**This is last line of template text**

**code:- (To display current date and time)**

**------**

**<%**

**java.util.Date now = new java.util.Date();**

**out.print(now);**

**%>**

**->out variable is called as an implicit variable.**

**what is implicit variable(or)object ?**

**-> The object which is created by web server automatically while creating the .java code of .jsp page.**

**They are: pageContext,application,config,session,out,exception,request,response etc.....**

**Note: These implicit variables can use directly as part of jsp pages without any declaration/definition.**

**Jsp Directives:**

**---------------**

**->A directive is the instruction given by the jsp developer to the jsp compiler.The jsp compiler generates the servlet's code according to the directive/instruction.**

**They are available in 3 types:**

**i) page ii) include iii) taglib.**

**page directive:-**

**--------------**

**code:- <%@ page import="java.util.Date" %>**

**here import = attribute name**

**java.util.Date is value of attribute.**

**->Current Date and Time:-**

**<%**

**Date now = new Date();**

**out.print(now);**

**%>**

**Note:-**

**We can use multiple page direcives to import multiple packages.**

**ex:- <%@ page import="java.util.\*" %>**

**<%@ page import="java.io.\*" %>**

**->we can use a single page directive for imporing multiple packages as shown below:**

**ex:- <%@page import="java.util.\*,java.io.\*,...." %>**

**error handling using JSPs:-**

**-------------------------**

**We can implement a separate setup pages to handle the errors.These pages are called as error pages.**

**->As part of the error page,A variable 'exception' can be used without declaring.This variable is called as implicit variable.**

**code:- (eh.jsp)**

**<%@page isErrorPage="true" %>**

**Our application failed due to:**

**<%**

**out.println(exception);**

**%>**

**<br>**

**Please send error info to admin @abc.com.**

**code:- (at.jsp)**

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

**<%@page errorPage="eh.jsp" %>**

**Trying to copy...**

**<%**

**java.io.FileInputStream fis = new java.io.FileInputStream("one.txt");**

**//code to open two.text**

**//code to copy info from one.txt to two.txt.**

**%>**

**--------**

**-> As part of jsp's scriptlet, we can use java and javascript also.**

**-> But this is possible in only one server: resine**

**ex:-**

**<%@page language="javascript" %>**

**<%**

**javascript code;**

**%>**

**<%@page language="java" %>**

**<%**

**java code;**

**%>**

**--------**

**ex:- (example for buffering mechanism)**

**<%@page autoFlush="false" buffer="300kb" %>**

**<%**

**for(int i=1;i<=100000;i++){**

**out.println("x");**

**}**

**%>**

**Difference between JspWriter and PrintWriter:-**

**--------------------------------------------**

**1.The print method of JspWriter throws IOException to report an error but the print() methods of PrintWriter are not designed to throw an IOException.**

**2.JspWriter supports Buffering mechanism but this is not supported by the PrintWriter.**

**We can use one of the following two solutions for solving the buffer overflow problem:**

**-->By increasing the Buffer size to accomodate the output that is produced by the jsp completely.**

**-->By setting autoflush to true.In this case the buffer will be flushed when it is full(sending the output available in the buffer to the client and clearing the buffer is called as flushing)**

**JSP Expressions:**

**----------------**

**syntax:**

**<%=... %>**

**-> These are used at outside scriptlet.**

**-> These are used to send data to the browser.so they are equivalent to JspWriter object.**

**-> By using these we can access scriptlet data from outside scriptlet**

**ex:- 1:**

**<%**

**int x = 10;**

**out.println("value of x from scriptlet:" + x);**

**%>**

**<br>**

**Value of x from template text:<%=x %>**

**ex:- 2:**

**OurCls.java:-**

**------------**

**package org.students;**

**import java.util.\*; //Date**

**public class OurCls**

**{**

**public int add(int a,int b){**

**return (a+b);**

**}**

**public Date getDate(){**

**return new Date();**

**}**

**public void mone(){**

**System.out.println("inside mone");**

**}**

**public String toString(){**

**return "This is OurCls object";**

**}**

**}**

**/\***

**compiling above program:**

**cmd:\>javac -d . OurCls.java**

**note:- copy "org" folder into classes folder of our project**

**toString():-**

**---------**

**This is mainly used in application debugging.**

**This method is called by JVM automatically when we are creating OurCls object.**

**\*/**

**using OurCls features in test.jsp:-**

**----------------------------------**

**<%@page import="org.students.\*" %>**

**<%**

**OurCls obj = new OurCls();**

**%>**

**<%=obj %><br>**

**Addition is:<%=obj.add(100,200) %><br>**

**Current Date and Time:<%=obj.getDate() %><br>**

**<%=obj.mone() %> (this line is not working here)**

**------------------------------**

**Jsp Declarations:**

**-----------------**

**syntax:**

**<%!**

**statements;**

**%>**

**-> these are stored outside \_jspService() method developed by jsp compiler.**

**-> so we can't use implicit variables directly as part of this.**

**-> But we can use implicit variables by passing as argument for any method which is available in declarations from scriptlet.**

**-> They are used to define our own jspinit() and jspdestroy()methods.**

**Note: As part of declarations,we can declare any type of data.**

**ex:- 1: (test.jsp)**

**<%!**

**int x = 10;**

**static int y = 20;**

**public void mone(JspWriter out)throws Exception{**

**out.println("<br>inside mone()");**

**}**

**public static void mtwo(){**

**System.out.println("inside mtwo()");**

**}**

**%>**

**<%**

**int z = 30;**

**out.println("Value of x :" + x);**

**out.println("<br>Value of y:" + y);**

**out.println("<br>Value of z:" + z);**

**mone(out);**

**mtwo();**

**%>**

**ex:- 2:(test.jsp)**

**<%!**

**public void jspInit(){**

**System.out.println("init called...");**

**}**

**public void jspDestroy(){**

**System.out.println("destroy called..");**

**}**

**%>**

**<%**

**System.out.println("service called..");**

**%>**

**Note:**

**configuring JSP File in web.xml:-**

**ex:-**

**<web-app>**

**<servlet>**

**<servlet-name>sone</servlet-name>**

**<jsp-file>/test.jsp</jsp-file>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>sone</servlet-name>**

**<url-pattern>/sone</url-pattern>**

**</servlet-mapping>**

**</web-app>**

**JSP Action tags and Java Beans:**

**-------------------------------**

**Action Tags:-**

**------------**

**-> They are used to add the dynamic behaviour to the JSP page**

**-> Depends upon the given tag, jsp container will generates the appropriate java code at runtime.**

**-> JSP having set of Action Tags:**

**some of the tags are:**

**<jsp:include>**

**<jsp:forward>**

**<jsp:param>**

**<jsp:useBean>**

**<jsp:setProperty>**

**<jsp:getProperty>**

**note: here jsp is nothing but 'prefix'.**

**<jsp:include> :- (dynamic include)**

**This is used to send the current page request to the another page. and includes the resultant output of another page into current page.**

**syntax:-**

**<jsp:include attributes>**

**<jsp:param attributes>**

**</jsp:include>**

**attributes:-**

**page: relative path of another resource(ex:- /hai.jsp)**

**flush: true/false (by default : false)**

**Note: this is flexible in application development.**

**Example using <jsp:include><jsp:forward><jsp:param> tags:-**

**---------------------------------------------------------------------------------**

**index.html:-**

**<HTML>**

**<HEAD>**

**<TITLE> My Page </TITLE>**

**</HEAD>**

**<BODY>**

**<BODY bgcolor=lightblue>**

**<form action="controller.jsp" method="post">**

**<center>**

**<table border=1>**

**<tr>**

**<td>Operand 1 :</td>**

**<td><input type="text" name="fone"></td>**

**</tr>**

**<tr>**

**<td>Operand 2 :</td>**

**<td><input type="text" name="ftwo"></td>**

**</tr>**

**<tr>**

**<td align=center><input type="submit" name="Submit" value="ADD" ></td>**

**<td align=center><input type="submit" name="Submit" value="SUB" ></td>**

**</tr>**

**<table>**

**</center>**

**</form>**

**</BODY>**

**</HTML>**

**controller.jsp:-**

**<%**

**String fone = request.getParameter("fone");**

**String ftwo = request.getParameter("ftwo");**

**String butt = request.getParameter("Submit");**

**if( butt.equals("ADD") ){**

**%>**

**<jsp:forward page="/addJsp.jsp">**

**<jsp:param name="x" value="<%=fone %>" />**

**<jsp:param name="y" value="<%=ftwo %>" />**

**</jsp:forward>**

**<%**

**}else if( butt.equals("SUB") ){**

**%>**

**<jsp:forward page="/subJsp.jsp">**

**<jsp:param name="x" value="<%=fone %>" />**

**<jsp:param name="y" value="<%=ftwo %>" />**

**</jsp:forward>**

**<%**

**}**

**%>**

**addJsp.jsp:-**

**<%**

**int a = Integer.parseInt(request.getParameter("x"));**

**int b = Integer.parseInt(request.getParameter("y"));**

**int res = a + b;**

**%>**

**<jsp:forward page="/result.jsp">**

**<jsp:param name="res" value="<%=res %>" />**

**</jsp:forward>**

**subJsp.jsp:-**

**<%**

**int a = Integer.parseInt(request.getParameter("x"));**

**int b = Integer.parseInt(request.getParameter("y"));**

**int res = a - b;**

**%>**

**<jsp:forward page="/result.jsp">**

**<jsp:param name="res" value="<%=res %>" />**

**</jsp:forward>**

**result.jsp:-**

**<%**

**String butt = request.getParameter("Submit");**

**String res = request.getParameter("res");**

**if( butt.equals("ADD") ){**

**%>**

**<center>Addition of 2 numbers:<%=res %><br>**

**<%**

**}else if( butt.equals("SUB") ){**

**%>**

**<center>Subtraction of 2 numbers:<%=res %><br>**

**<%**

**}**

**%>**

**<jsp:include page="/index.html" />**

**---------------------------------------------------------------------------------------------------------------------**

**Difference b/w include tag and include directive:-**

**-------------------------------------------------**

**ex:- (one.jsp)**

**output of one.jsp<br>**

**<jsp:include page="/two.jsp" />**

**two.jsp:-**

**output of two.jsp**

**-> here two .java and .class files are created at runtime**

**-> These two .class files are execute and sends the output to the client.**

**include directive:- (static include)**

**-----------------**

**syntax:**

**<%@include attributes %>**

**ex:- (one.jsp)**

**output of one.jsp<br>**

**<%@include file="/two.jsp" %>**

**two.jsp:-**

**output of two.jsp**

**-> here include directive merges the output of two.jsp into one.jsp and creates a .java file at compiletime.**

**Note: This is used in fast accessing of multiple jsp's.**

**<jsp:forward>:-**

**This is used to forward the current page request to another page. And sends the resultant output of forwarded page only.**

**<jsp:forward attributes>**

**<jsp:param attributes>**

**</jsp:forward>**

**Note: In the above two tags we can use <jsp:param> to provide the name and value pair to the second jsp at runtime.**