**Q) What is not JSP?**

* JSP is not software to be installed.
* JSP is not a programming language unlike Java.

**Q) What is JSP?**

* JSP is a Java based Web Technology.
* JSP is a J2EE technology.

**Q) What is the purpose of JSP technology?**

* JSP technology is used in Java based web application development.

OR

* JSP technology is used in the implementation of Java based web-enabled enterprise applications.

OR

* JSP technology is used in Java based dynamic website development.

**Q) What is a jsp?**

* A jsp is a web server side piece of code that enhances the functionality of the web server.
* A jsp is a dynamic web resource in a Java web application.
* A jsp is a web component.

**Q)What can a jsp do in making a Java based website dynamic?**

* A jsp can perform the following tasks.

1. capturing the user input
2. communicating with the database
3. processing of data
4. producing the response page
5. handing over the response page to the web server

**Q) What are the constituents of a Jsp?**

* Jsp= HTML code+ Java code(directly/indirectly)

**Q) Explain about the life cycle of a Jsp.**

* JSP container/engine controls the life of a JSP page (Jsp).
* Life cycle of a Jsp is described by3 life cycle methods and 6 phases.
* Life cycle methods are

1. jspInit()
2. jspService(HttpServletRequest,HttpServletResponse)
3. jspDestroy()

* Life cycle phases are

1. translation phase
2. compilation phase
3. instantiation phase
4. initialization phase
5. servicing phase
6. destruction phase

**Q) Explain about translation phase of a Jsp.**

* JSP Container converting ".jsp" file into ".java" file is nothing but translation phase of a jsp.
* A jsp(JSP page) is a pure text file with any name but with ".jsp" extension.
* A jsp contains HTML code but it is not a HTML document.
* A jsp can contain Java code but it is not a Java program.
* Whenever client request comes for the first time for a jsp, JSP container translates the jsp into a Java file. This Java file in fact, is the source code of a servlet. I.e. during the translation phase of a jsp, JSP engine converts it into a servlet. This servlet is known as "Container generated servlet"OR "Page Implementation Class".
* Unless the source code is modified, translation happens only once in the lifetime of a jsp.

**Q) Explain about compilation phase of a jsp.**

* JSP Container compiling the page implementation class of a jsp is nothing but its compilation phase.
* Compilation happens only once unless the source code of the jsp is modified.
* JSP Container uses time stamp to know whether the jsp is modified or not.

**Q) Explain about instantiation, initialization, servicing and destruction phases of a jsp.**

* Container instantiating the page implementation class is nothing but instantiation phase of the Jsp.
* Container calling jspInit () method of the page implementation class is nothing but initialization phase of the jsp.
* Instantiation and initialization happens only once in the life time of a jsp.
* jsp developer(page author) provides any resources required for the jsp in jspInit() method. For eg.database connection creation &PreparedStatementcreation.
* Container calling jspService() method is nothingbut servicing phase of the jsp. For each client request Container calls jspService() method.
* JSP container is responsible to generate jspService() method.
* In a jsp, jsp developer should not implement jspService() method. If required, jspInit() &jspDestroy() can be implemented.
* When application is unloaded or Container is shutdown, container calls jspDestroy() and marks the page implementation class instance for garbage collection.
* Container calling jspDestroy() method is nothingbut destruction phase of the jsp.
* Destruction happens only once in thelife time of a jsp.
* Resources if any allocated during initializationphase are deallocated in jspDestroy() method.

**Q)Explain about instantiation, Initialization,servicing and destruction phase of a jsp.**

* Container instantiating the page implementation class is nothing but instantiation phase of the jsp.
* Container calling jspInit () method of the page implementation class is nothing but initialization phase of the jsp.
* Instantiation and initialization happens only once in the life time of a Jsp.
* jsp developer(page author) provides any resources Required for the jsp in jspInit () method. For eg.Database connection creation &PreparedStatementcreation.
* Container calling jspService () method is nothing but servicing phase of the jsp. for each client request container calls jspService() method.
* JSP container is responsible to generate jspService () method.
* In a jsp, jsp developer should not implement jspService () method. If required, jspInit () &jspDestroy () can be implemented.
* When application is unloaded or Container is shutdown, container calls jspDestroy () and marks the page implementation class instance for garbage collection.
* Container calling jspDestroy () method is nothing but destruction phase of the jsp.
* Destruction happens only once in the life time of a jsp.
* Resources if any allocated during initialization phase are dealocated in jspDestroy () method.

**Q) Explain about translation phase of a jsp.**

* JSP Container converting a jsp into a servlet is nothing but translation phase of a jsp.
* When client request comes for a jsp for the first time after its deployment, JSP engine makes a full read of the source code of the jsp and verifies the syntactical correctness of the JSP elements(if any) used in the jsp. If JSP elements are syntactically correct, it converts ".jsp" file into ".java “file.
* ".java" file created during translation phase is nothing but the source code of a servlet.I.e. during translation phase of a jsp, JSP container generates a servlet. This servlet is known as container generated servlet OR page implementation class.

**Q) Explain about compilation phase of a jsp.**

* JSP Engine compiling the page implementation class source code to “.class" file is nothing but compilation phase of a jsp.

**Note: -** translation and compilation happens only once in the life time of a jsp unless its source code is modified.

* JSP Container uses Time Stamp to know whether the jsp is modified.

**Q) Explain about JSP elements.**

* JSP technology provided server side programming elements (constructs) using which, a jsp has dynamic functionality are nothing but JSP elements.
* JSP elements are of 3 types.

1. Scripting Elements
2. Directives
3. actions(tags)

**Q) What are scripting elements?**

* JSP scripting elements are those JSP elements using which, Java code is directly placed into (the source code of a jsp.
* There are 3 kinds of Scripting elements.

1. declaration
2. expression
3. scriptlet

* JSP declaration is used to place class scoped (both instance and static) variables and methods in a jsp.
* A JSP declaration starts with <%! and ends with %>

For eg.

<%!

int a;

void m()

{

}

%>

* During translation phase, JSP Container makes variables& methods of a JSP declaration as the members of the page implementation class.
* In a jsp, we can have any number of JSP declarations.

**Expression**

* JSP expression is one of the JSP Scripting elements.
* It is used to embed Java code directly into a jsp.
* A JSP expression starts with <%=and ends with %>
* In a JSP expression we can have only one Java expression which is required to be executed for every client request.

For eg.

1.) <%= a+b %>

2.) <%= rs.getFloat(3) %>

3.) <%= sum %>

* During translation phase, Java expression is placed into jsp Service method of the page implementation class.

**Note:-** We should not write semicolon at the end of Java expression within a JSP Expression.

* Two things happen when a JSP expression is executed.
  1. Java expression is evaluated.
  2. The result is written to the browser stream.
* We can have any number of JSP expressions in a jsp. During translation phase, in the same order Java code is placed into jspService method.

**Scriptlet**

* JSP Scriptlet is one of the JSP Scripting elements.
* It is used to embed Java code directly into a jsp.
* A JSP Scriptlet starts with <% and ends with %>
* In a JSP scriptlet we can have any Java code that we generally write in doGet/doPost method of a user defined servlet.

foreg.

<%

String user=request.getParameter("user");

String pwd=request.getParameter("password");

ps.setString(1,user);

ps.setString(2,pwd);

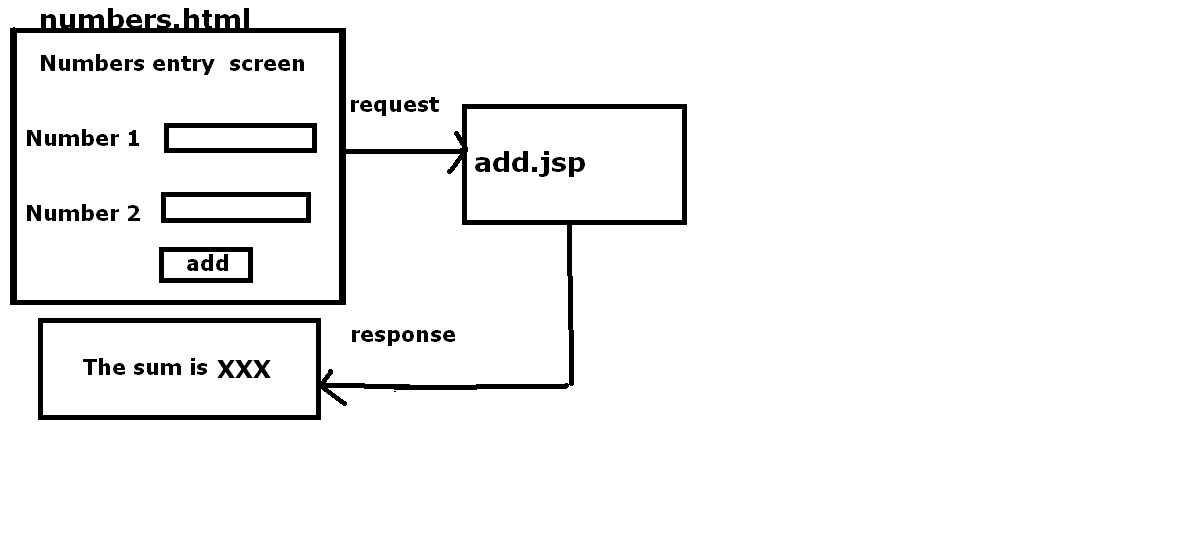
ResultSetrs=ps.executeQuery();

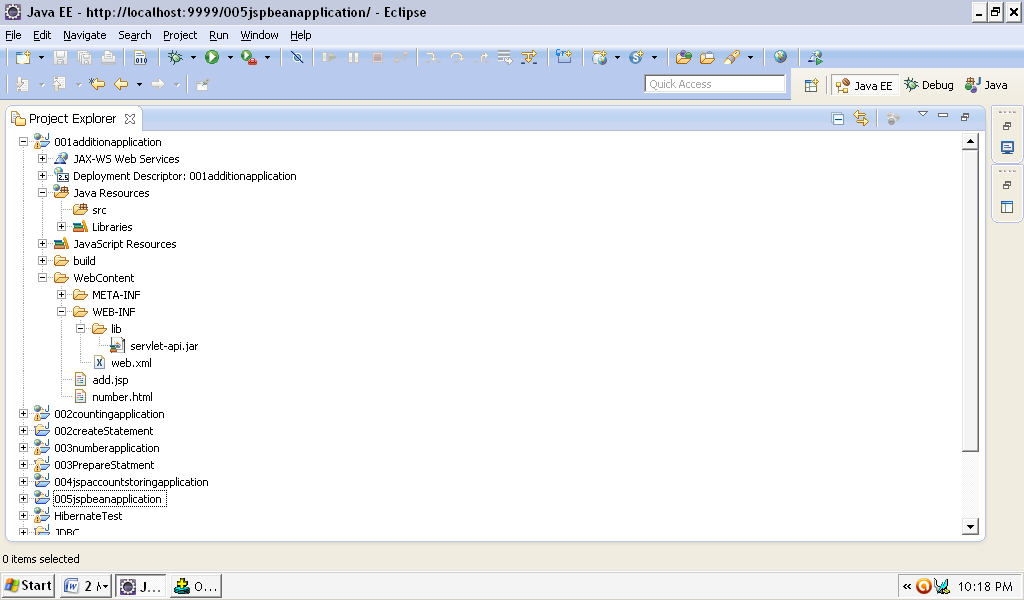
......

%>

* During translation phase, Java code written in a scriptlet is placed into jspService method of the page implementation class.
* We can have any number of JSP scriptlets in a jsp. During translation phase, in the same order Java code is placed into jspServicemethod.

**Q) Develop a Java web application to implement the use-case of addition of two numbers.**

****

****

**numbers.html**

1. <!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN"
2. "http://www.w3.org/TR/html4/loose.dtd">
3. <HTML>
4. <BODYBGCOLOR=*"yellow"*>
5. <CENTER>
6. <H1>Numbers entry screen</H1>
7. <FORMACTION=*"add.jsp"*>
8. Number 1<INPUTTYPE=*"text"*NAME=*"n1"*><BR>
9. <BR> Number 2<INPUTTYPE=*"text"*NAME=*"n2"*><BR>
10. <BR><INPUTTYPE=*"submit"*VALUE=*"add"*>
11. </FORM>
12. </CENTER>
13. </BODY>
14. </HTML>

**add.jsp**

1. <%@pagelanguage=*"java"*contentType=*"text/html; charset=ISO-8859-1"*
2. pageEncoding=*"ISO-8859-1"*%>
3. <!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN"
4. "http://www.w3.org/TR/html4/loose.dtd">
5. <%
6. **int** n1 = Integer.parseInt(request.getParameter("n1"));
7. **int** n2 = Integer.parseInt(request.getParameter("n2"));
8. **int** sum = n1 + n2;
9. %>
10. <HTML>
11. <BODYBGCOLOR=*"cyan"*>
12. the sum is<%=sum%>
13. </BODY>
14. </HTML>

**web.xml**

1. <?xmlversion=*"1.0"*encoding=*"UTF-8"*?>
2. <web-appxmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance*
3. xmlns=[*http://java.sun.com/xml/ns/javaee*](http://java.sun.com/xml/ns/javaee)
4. xmlns:web=[*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)
5. xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*
6. [*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)*"*
7. id=*"WebApp\_ID"*version=*"2.5"*>
8. <display-name>additionapplication</display-name>
9. <welcome-file-list>
10. <welcome-file>number.html</welcome-file>
11. </welcome-file-list>
12. </web-app>

**Note:**jsp need not be registered forthe JSP container to excute it.

**Q) What are the hidden comments in the context of a jsp?**

* JSP comments of a jsp are not visible at the client side in the response page unlike that of HTML and hence called as hidden comments.

For eg.

<%-- added and displayed --%>

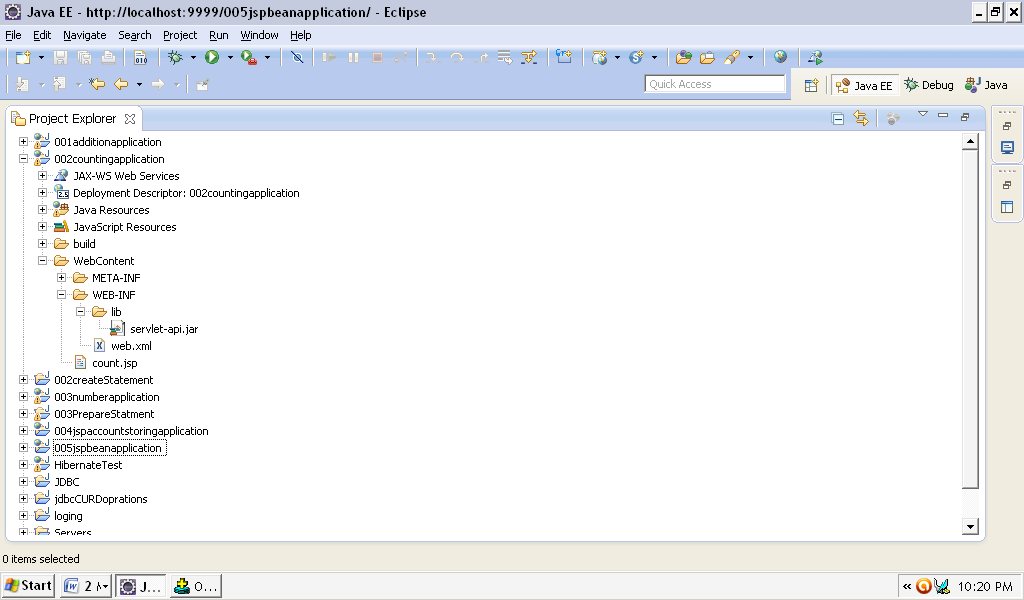
**Q) What is the default MIME type in a servlet(user defined servlet)?**

text/plain

**Q) What is the default MIME type in a jsp(container generated servlet)?**

text/html

**Q) Develop a Java web application that displays to the user every time the number of times that page is visited by that user.**

****

**count.jsp**

1. <html>
2. <head>
3. <title>Applcation object in JSP</title>
4. </head>
5. <body>
6. <%
7. Integer hitsCount = (Integer) application.getAttribute("hitCounter");
8. **if** (hitsCount == **null** || hitsCount == 0) {
9. /\* First visit \*/
10. out.println("Welcome to my website!");
11. hitsCount = 1;
12. } **else** {
13. /\* return visit \*/
14. out.println("Welcome back to my website!");
15. hitsCount += 1;
16. }
17. application.setAttribute("hitCounter", hitsCount);
18. %>
19. <center>
20. <p>
21. Total number of visits:
22. <%=hitsCount%>
23. </p>
24. </center>
25. </body>
26. </html>

**Web.xml**

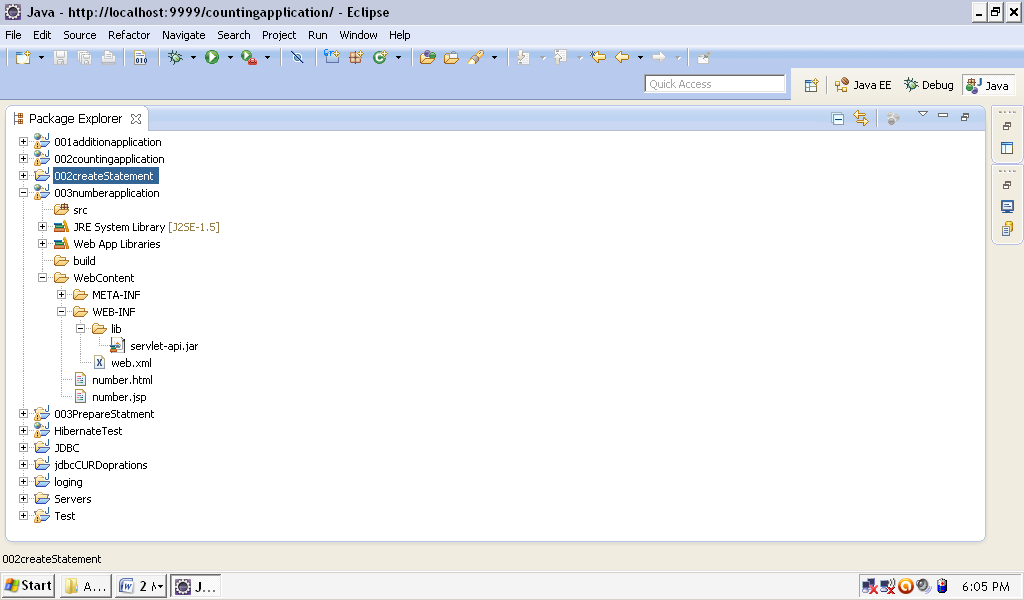
1. <?xmlversion=*"1.0"*encoding=*"UTF-8"*?>
2. <web-appxmlns:xsi=[*http://www.w3.org/2001/XMLSchema-instance*](http://www.w3.org/2001/XMLSchema-instance)
3. xmlns=[*http://java.sun.com/xml/ns/javaee*](http://java.sun.com/xml/ns/javaee)
4. xmlns:web=[*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)
5. xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*
6. [*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)*"*id=*"WebApp\_ID"*version=*"2.5"*>
7. <display-name>numberapplication</display-name>
8. <welcome-file-list>
9. <welcome-file>number.html</welcome-file>
10. </welcome-file-list>
11. </web-app>

**Q)Develop a Java web application in which, end-user enters a number into a web form and gets the following response.**

XXX is bigger than 100

OR

XXX is smaller than 100



**number.html**

1. <html>
2. <BODYBGCOLOR=*"yellow"*>
3. <CENTER>
4. <H1>Number entry screen</H1>
5. <FORMACTION=*"number.jsp"*>
6. Number <INPUTTYPE=*"text"*NAME=*"n"*><BR><BR>
7. <INPUTTYPE=*"submit"*VALUE=*"send"*>
8. </FORM>
9. </CENTER>
10. </BODY>
11. </html>

**number.jsp**

1. <HTML>
2. <BODYBGCOLOR=*"cyan"*>
3. <%
4. **int** n = Integer.parseInt(request.getParameter("n"));
5. **if** (n > 100) {
6. %>
7. <H1>
8. <%=n%>
9. is bigger than 100
10. </H1>
11. <%
12. } **else** {
13. %>
14. <H1><%=n%>
15. is smaller than 100
16. </H1>
17. <%
18. }
19. %>
20. </BODY>
21. </HTML>

**Web.xml**

Same as above

**Q) Explain about JSP directives.**

* JSP directives are one type of JSP elements.
* A JSP directive is a translation time instruction to the JSP container. It is also known as static instruction to the container.
* There are 3 types of directives.

1. include
2. page
3. taglib

* A JSP directive has the following syntax.

<%@ directivename attribute="value" .... %>

**Q) Explain about include directive.**

* JSP include directive is used to implement include mechanism in a jsp.
* JSP include mechanism is meant for reusing the presentation code (output generation code) across multiple pages of the website.

For e.g.

some.jsp

-----------

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

<! -- body of some.jsp goes here -->

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

**Note:-**some.jsp is known as including page. header.jsp& footer.html are known as included pages

* When JSP engine encounters the include directive in the including page, it copies the source code of the included page in line in the including page and then it translates the including page into a servlet.
* In case of include directive, inter-jsp communication doesn't happen.

**Note:-**include directive has a limitation. If included page content is modified at later stage of application running, modified output will not be reflected in the

response of the including page.

**Q) Explain about taglib directive.**

* taglib directive is used to import custom tag libraries into a jsp.
* It has the following syntax.

<%@ taglib prefix="alias name of the custom tag library"

uri="actual name of the custom tag library" %>

**Q) Explain about page directive.**

* page directive is used to specify the structure of page implementation class to the JSP container.
* page directive has 13 attributes.
  + 1. import
    2. errorPage
    3. isErrorPage
    4. language
    5. contentType
    6. session
    7. extends
    8. isELIgnored
    9. autoFlush
    10. info
    11. pageEncoding
    12. buffer
    13. isThreadSafe

**import**

* import attribute of page directive is used to import a Java package into a jsp.

For eg.

<%@ page import="java.sql.\*,java.util.\*" %>

OR

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

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

**Note:-** By default, in every jsp 4 java packages are implicitly available.

* + 1. javax.servlet
    2. javax.servlet.http
    3. javax.servlet.jsp
    4. java.lang

**isErrorPage**

* By default, a jsp is not an error page. i.e. exception handling page. If we want to designate a jsp as error page, we use "isErrorPage" attribute by giving "true" as value. Default value is "false".

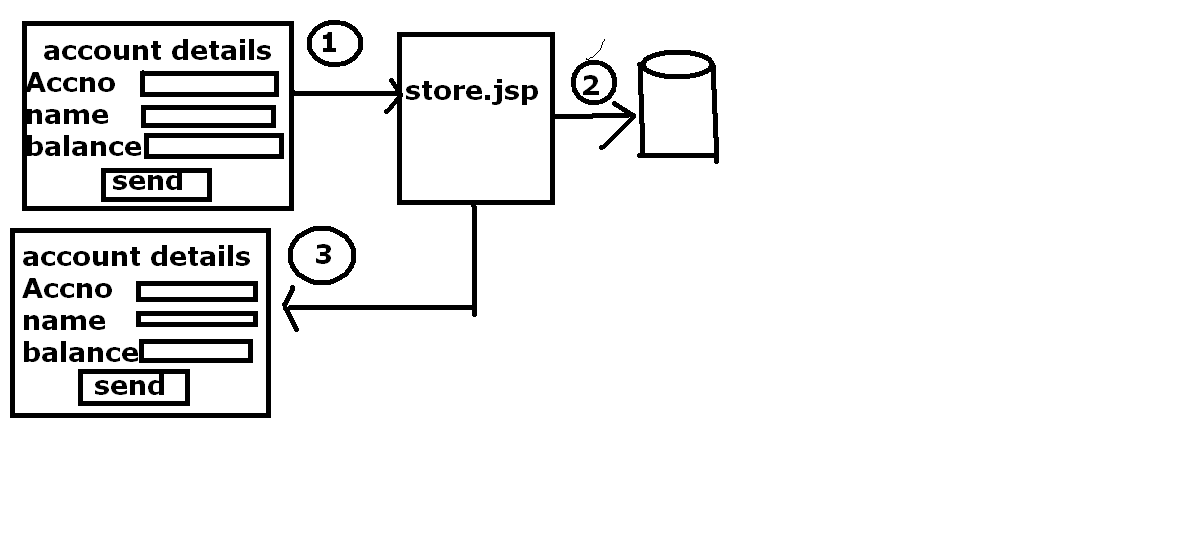
**errorPage**

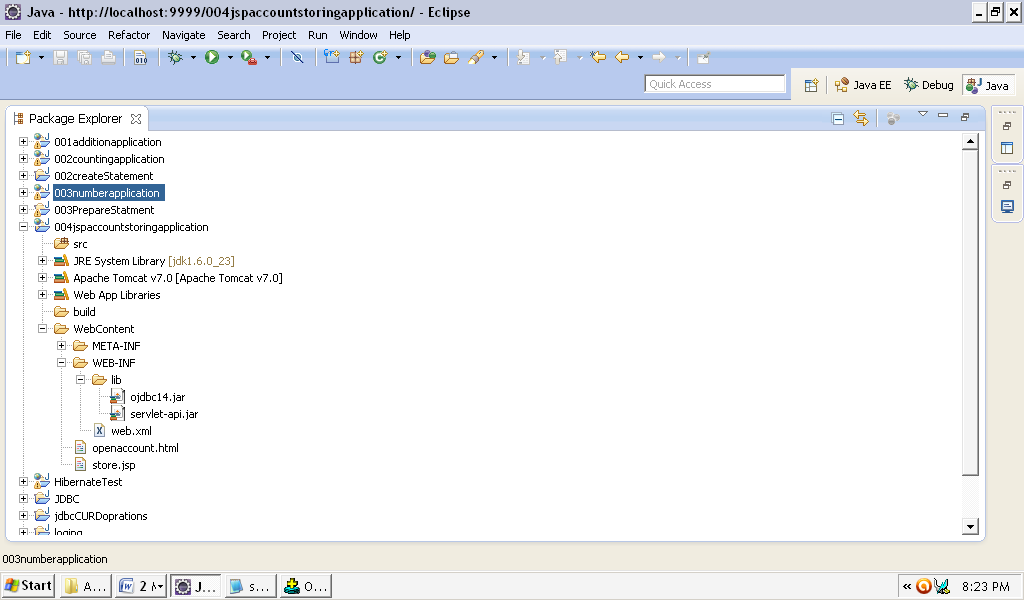
* This attribute is used to associate a normal jsp to an error page so that if any exception is raised in that page control goes to the error page and the exception is handled.

**Q) Develop a Java web application in which, account storing(into the database) use case is implemented(using jsp).**

**Jspaccountstoringapplication**

**accountsotringapplications.bmp**

****

****

**openaccount.html**

1. <html>
2. <head>
3. <meta http-equiv=*"Content-Type”* content=*"text/html; charset=ISO-8859-1"*>
4. <title>Insert title here</title>
5. </head>
6. <body bgcolor=*"wheat"*>
7. <CENTER>
8. <H1>A/c details</H1>
9. <FORMACTION=*"store.jsp"*METHOD=*"POST"*>
10. ACCNO<INPUTTYPE=*"text"*NAME=*"accno"*><BR><BR>
11. NAME<INPUTTYPE=*"text"*NAME=*"name"*><BR><BR>
12. BALANCE<INPUTTYPE=*"text"*NAME=*"balance"*><BR><BR>
13. <INPUTTYPE=*"submit"*VALUE=*"store"*>
14. </FORM>
15. </CENTER>
16. </body>
17. </html>

**store.jsp**

1. <%@ page import=*"java.sql.\*"*%>
2. <html>
3. <head>
4. <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>
5. <title>Insert title here</title>
6. </head>
7. <body>
8. <%!Connection con;
9. PreparedStatement ps;
10. **Publicvoid** jspInit() {
11. **try** {
12. Class.forName("oracle.jdbc.driver.OracleDriver");
13. con = DriverManager
14. .getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
15. "scott", "tiger");
16. ps = con.prepareStatement("INSERT INTO ACCOUNT VALUES(?,?,?)");
17. } **catch** (Exception e) {
18. System.out.println(e);
19. }
20. }//jspInit()
21. **Publicvoid** jspDestroy() {
22. **try** {
23. ps.close();
24. con.close();
25. } **catch** (SQLException e) {
26. System.out.println(e);
27. }
28. }%>
29. <%
30. **Int** ano = Integer.parseInt(request.getParameter("accno"));
31. String nm = request.getParameter("name");
32. **Float** bal = Float.parseFloat(request.getParameter("balance"));
33. ps.setInt(1, ano);
34. ps.setString(2, nm);
35. ps.setFloat(3, bal);
36. ps.executeUpdate();
37. System.out.println("account successfully created");
38. %>
39. <%@include file=*"openaccount.html"*%>
40. </body>
41. </html>

**web.xml**

1. <?xml version=*"1.0"* encoding=*"UTF-8"*?>
2. <web-app xmlns:xsi=[*http://www.w3.org/2001/XMLSchema-instance*](http://www.w3.org/2001/XMLSchema-instance)
3. xmlns=[*http://java.sun.com/xml/ns/javaee*](http://java.sun.com/xml/ns/javaee)
4. xmlns:web=[*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)
5. xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*
6. [*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*](http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd)*"*
7. id=*"WebApp\_ID"* version=*"2.5"*>
8. <display-name>004jspaccountstoringapplication</display-name>
9. <welcome-file-list>
10. <welcome-file>openaccount.html</welcome-file>
11. </welcome-file-list>
12. </web-app>

**Q) Explain about JSP implicit objects.**

* In a jsp, 9\* objects(references) are implicitly available and are known as JSP implicit objects.

|  |  |
| --- | --- |
| Request | HttpServletRequest |
| response | HttpServletResponse |
| Config | ServletConfig |
| Session | HttpSession |
| application | ServletContext |
| Out | javax.servlet.jsp.JspWriter |
| page | java.lang.Object |
| exception | java.lang.Throwable |
| pageContext | javax.servlet.jsp.PageContext |

* During translation phase of a jsp, JSP container declares these 9\* references in jspService () method and initializes them. Therefore, JSP implicit objects are not available in declaration section of a jsp.Theyare available only in a JSP Expression and a JSP scriptlet.

**Q) Explain about "out" implicit object.**

* "out" is a reference to the instance of JspWriter class of JSP API. It is a sub class of java.io.Writer.
* It is the browser stream for a jsp. i.e.Whatever is written into this stream is sent to the browser.
* JspWriter is similar to that of PrintWriter of user defined servlet with additionalbuffering capabilities.
* This implicit object is the most widely usedone of all. But,invisibly. Almost all the cases, it is not directly used in a jsp. Therefore, it isknown as invisible work horse.

**Q) Explain about "page" implicit object.**

* "page" is one of the 9 JSP implicit objects.
* "page" is a local variable in jspService()method of type java.lang.Object.
* "page" reference variable holds the referenceof (hashCode) page implementation classobject.
* This implict object is seldom used.

**Q) Explain about "exception" implicit object.**

* "exception" is one of the 9 JSP implicit objects.
* "exception" is a local variable in jspService() method of type java.lang.Throwable.
* This implicit object is not available in all jsps. It is available only in those jsps which are designated as error pages using "isErrorPage" attribute of page directive.
* When a processing page\* is associated with an error page, if abnormal event occurs in processing page, that abnormality details are wrapped into "exception" and passed to the error page wherein, it is handled.

**Q)What are JSP actions(Tags)?**

* A JSP tag is a request processing time(runtime/dynamic) instruction to the JSP Container.
* There are two kinds of JSP actions(tags).
  + 1. standard tags
    2. custom tags

**Q) Explain about standard tags of JSP.**

* JSP standard tags are the pre-defined tags given by JSP technology whose meaning is already made known to the JSP container.
* Following are the standard actions of JSP.
  1. include
  2. forward
  3. param
  4. useBean
  5. setProperty
  6. getProperty

**include**

* includestandard tag is used to implement include mechanism in a jsp.
* include standard tag overcomes the limitationof include directive at the cost of slight performance.

For eg.

some.jsp

-----------

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

* The response generated by headerjsp is included in line in the response generated by some.jsp.
* With include tag, inter jsp communication is implemented. Where as with include directive it is not.

**forward**

* forward tag is used to implement inter-jspcommunication through forward mechanismof request dispatching.

For eg.

source.jsp

------------

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

source.jsp

-------------

<%

pageContext.forward("target.jsp");

%>

OR

source.jsp

-------------

<%

RequestDispatcherrd=request.getRequestDispatcher("target.jsp");

rd.forward(request,response);

%>

**Param**

* "param" standard tag is used to pass/supply request parameters from one jsp to the other during inter-jsp communication.

foreg.

source.jsp

-------------

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

<jsp:param name="user" value="scott" />

<jsp:param name="pwd" value="tiger" />

</jsp:forward>

target.jsp

------------

<%

String username=request.getParameter("user");

String password=request.getParameter("pwd");

%>

**Q)Explain about Java Bean.**

* A Java Bean is a specialized Java class thatis implemented by following the rules specified in Java Beans specification.
* According to Java Beans specification, a Java class becomes a Java Bean if it is developed by observing the following rules.
  + 1. class is public
    2. class implements Serializable interface
    3. properties are private
    4. each property has one public setter method and public getter method
    5. class has public zero argument constructor

For eg.

public class Account implements java.io.Serializable

{

privateintaccno;

private String name;

private float balance;

public void setAccno(intano)

{

accno=ano;

}

publicintgetAccno()

{

returnaccno;

}

public void setName(String name)

{

this.name=name;

}

public String getName()

{

return name;

}

public void setBalance(float bal)

{

balance=bal;

}

public floatgetBalance()

{

returnthis.balance;

}

}//Java Bean

**Q)Explain about useBean,setProperty&getPropery standard actions.**

**useBean**

* useBean tag is used to instruct the JSPengine to instantiate a Java Bean(and/or getits reference).

For eg.

<jsp:useBean id="acc" class="Account" />

**setProperty**

* This tag is used to populate the bean fields.

For eg.

<jsp:setProperty name="acc" property="accno" value="1001" />

<jsp:setProperty name="acc" property="name" value="David" />

<jsp:setProperty name="acc" property="balance" value="10000" />

**Note:-** If data is coming from the web formand if request parameter names andbean field names are the same, we can say

<jsp:setProperty name="acc" property="\*" />

**getProperty**

* This tag is used to retrieve bean field valueand display to the user.

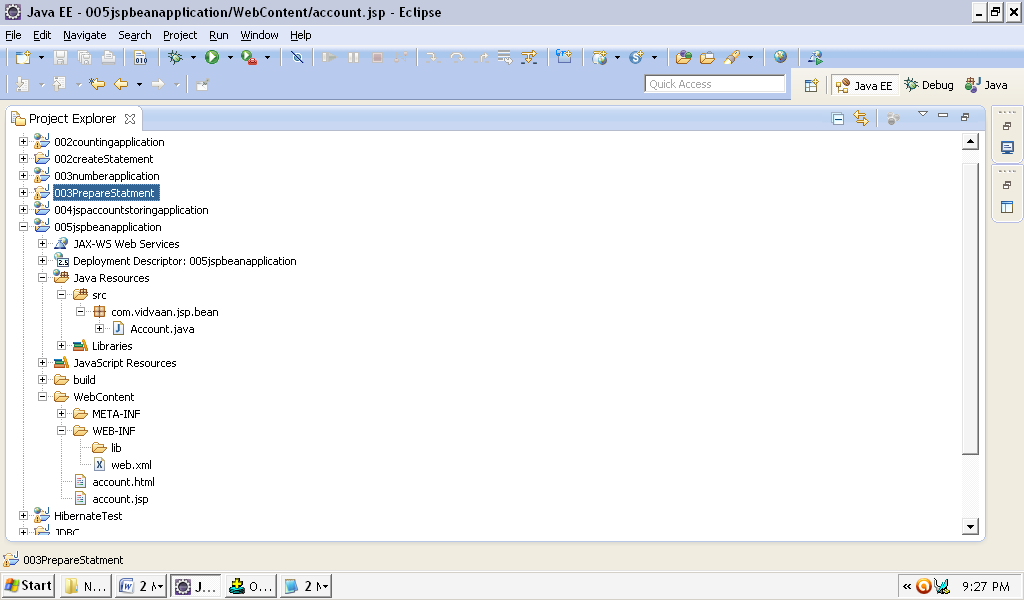
For eg.

<jsp:getProperty name="acc" property="accno" />

<jsp:getProperty name="acc" property="name" />

<jsp:getProperty name="acc" property="balance" />

**Q) Develop a Java web application in which, a Java Bean is used in a jsp.**

****

**Account.jsp**

1. **package**com.vidvaan.jsp.bean;
2. **publicclass** Account {
3. **privateint**accno;
4. **private** String name;
5. **privatefloat**balance;
6. **publicint**getAccno() {
7. **return**accno;
8. }
9. **publicvoid**setAccno(**int**accno) {
10. **this**.accno = accno;
11. }
12. **public** String getName() {
13. **return**name;
14. }
15. **publicvoid**setName(String name) {
16. **this**.name = name;
17. }
18. **publicfloat**getBalance() {
19. **return**balance;
20. }
21. **publicvoid**setBalance(**float** balance) {
22. **this**.balance = balance;
23. }
24. }

**account.html**

1. <html>
2. <head>
3. <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>
4. <title>Insert title here</title>
5. </head>
6. <body>
7. <CENTER>
8. <H1>A/c details</H1>
9. <FORMACTION=*"account.jsp"*>
10. ACCNO<INPUT TYPE=*"text"*NAME=*"accno"*><BR><BR>
11. NAME<INPUT TYPE=*"text"*NAME=*"name"*><BR><BR>
12. BALANCE<INPUT TYPE=*"text"*NAME=*"balance"*><BR><BR>
13. <INPUT TYPE=*"submit"*VALUE=*"store"*>
14. </FORM>
15. </CENTER>
16. </body>
17. </html>

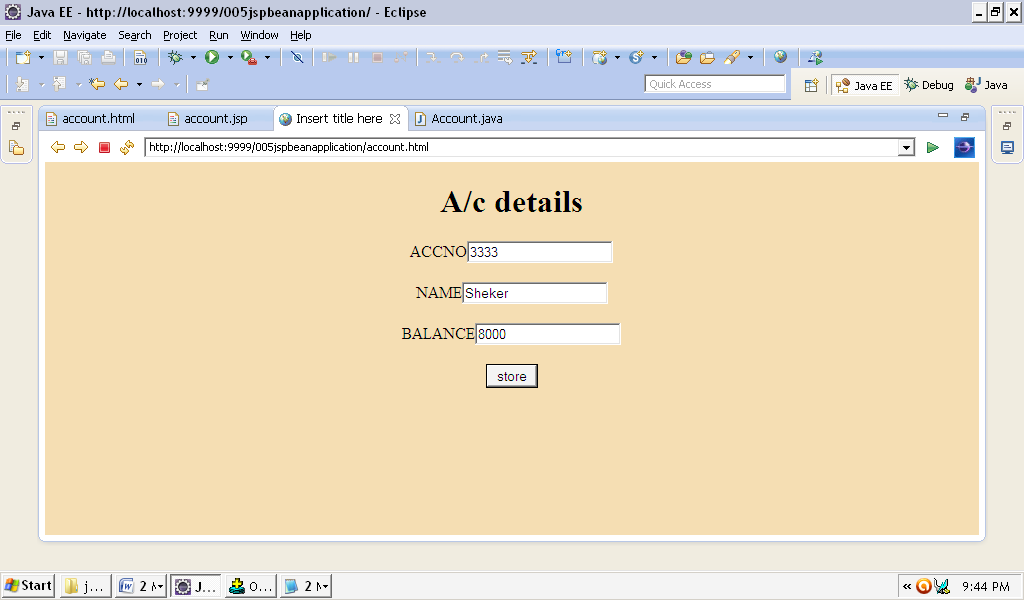
**account.jsp**

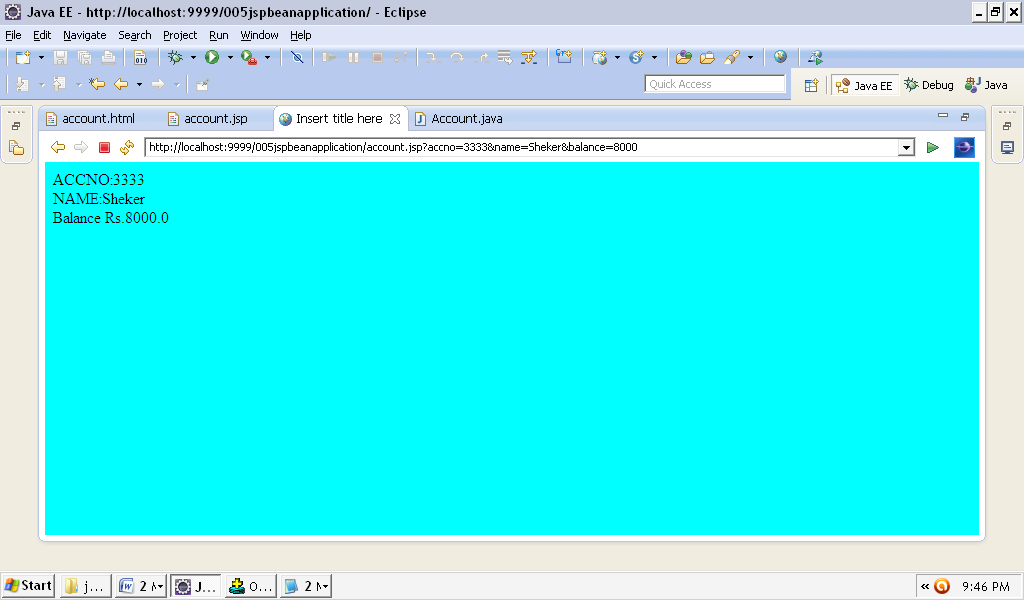
1. <jsp:useBeanid=*"acc"*class=*"com.vidvaan.jsp.bean.Account"*/>
2. <jsp:setPropertyname=*"acc"*property=*"\*"*/>
3. <html>
4. <head>
5. <metahttp-equiv=*"Content-Type"*content=*"text/html; charset=ISO-8859-1"*>
6. <title>Insert title here</title>
7. </head>
8. <bodybgcolor=*"cyan"*>
9. ACCNO:<jsp:getPropertyname=*"acc"*property=*"accno"*/><BR>
10. NAME:<jsp:getPropertyname=*"acc"*property=*"name"*/><BR>
11. Balance Rs.<jsp:getPropertyname=*"acc"*property=*"balance"*/>
12. </body>
13. </html>

**Web.xml**

//Same as aboue

See the out put





**Note:-** Java Bean was once used as a modelcomponent in an MVC based Java web application.

* In a Java bean normal methods also can be written

**Q)What are the attributes of page directive?**

* Page directive has 13 attributes.

1.import

2. errorPage

3. isErrorPage

4. language

5. contentType

6. session

7.extends

8.isELIgnored

9.autoFlush

10.info

11.pageEncoding

12.buffer

13.isThreadSafe

**Q)Explain about language attribute of page directive.**

* "language" directive is used to specify the language used within the scripting elements. By default, its value is "java".
* No other value is allowed till date(upoto JSP 2.2 specification).

<%@ page language="java" %>

* The above directive is redundantto write in a jsp.

**Q)Explain about "contentType" attribute.**

* This attribute is used to specify the MIME type to the container. Default value is "text/html"
* We can change the default value as follows.

<%@ page contentType="image/gif" %>

**Q) Explain about "session" attribute.**

* This attribute is used to specify the JSP engine whether the current jsp participates in session tracking or not. Default value is"true". We can change it as follows.

<%@ page session="false" %>

* Now, in the jsp "session" implicit object is not available and data stored in session object also not available. i.e. session scope is not available.

**Q)Explain about "extends" attribute.**

* "extends" attribute is used to specify to the JSP engine to generate the page implementation class so as not to inherit from its specific class; instead from a user defined class.
* To customize the functionality of the container, this attribute is used.
* This attribute is seldom used.

<%@ page extends="com.vidvaan.MyClass" %>

**Q)Explain about "isELIgnored" attribute of page directive**

* By default this attribute value is "false". i.e. JSP engine tries to evaluate the expression of Expression Language in the jsp by default.
* To instruct the JSP engine to treat that syntax as template text(HTML content) but not as EL syntax, we give "true" as value to this attribute. i.e. to disable EL in a jsp, this attribute is used.

<%@ pageisELIgnored="true" %>

**Q)Explain about "buffer" and "autoFlush"attributes.**

* "buffer" attribute specifies the minimum size required by the output buffer that holds the generated content until it is sent to the client.
* Specification specifies the minimum size of the buffer as 8kb.
* We can change its value as follows.

<%@ page buffer="32kb" %>

* The suffix "kb" is mandatory.
* "autoFlush" specifies whether the data in the output buffer should be sent to the client automatically as soon as the buffer is full. "true" is the default value.
* If we specify the "false", unless we say flush() on the "out" object, the content will not be sent to the client.

**Q)Explain about "info" attribute.**

* It is used to specify general purpose information about the jsp.
* page author, date of development etc. are specified using this attribute.

<%@ page info="This page developed by XYZ on 11/07/11" %>

* Programmatically, the value of this attribute can be retrieved by calling getServletInfo() method of Servlet interface.

**Q) Explain about "pageEncoding" attribute.**

* This attribute specifies the characterset type used in a jsp. By default, its value is "ISO-8859-1". We can change it as follows.

<%@ page pageEncoding="UTF-16" %>

**Q)Explain about "isThreadSafe" attribute.**

* This attribute is used to indicate to the JSP engine whether the page implementation class has to implement javax.servlet.SingleThreadModel interface or not.
* By default, its value is "true". It indicates that container generated servlet doesn't inherit from "SingleThreadModel" interface.

<%@ pageisThreadSafe="false" %>

* The above code instructs the JSP engine to generate the page implementation class so as to inherit from javax.servlet.SingleThreadModel interface.