1. What is JSP
JavaServer Pages. A server-side technology, JavaServer pages are an extension to the Java servlet technology that was developed by Sun. JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements -- the actual design and display of the page. Embedded in the HTML page, theJava source code and its extensions help make the HTML more functional, being used in dynamic database queries, for example. JSPs are not restricted to any specific platform or server.

2. What are advantages of JSP
whenever there is a change in the code, we dont have to recompile the jsp. it automatically does the compilation. by using custom tags and tag libraries the length of thejava code is reduced.

3. What is the difference between include directive & jsp:include action
should use the include directive ():
if the file includes static text
if the file is rarely changed (the JSP engine may not recompile the JSP if this type of included file is modified)
if you have a common code snippet that you can reuse across multiple pages (e.g. headers and footers)

should use the jsp:include
for content that changes at runtime
to select which content to render at runtime (because the page and src attributes can take runtime expressions)
for files that change often

4. What are Custom tags. Why do you need Custom tags. How do you create Custom tag

Custom tags are those which are user defined.

2) Inorder to separate the presentation logic in a separate class rather than keeping in jsp page we canuse custom tags.

3)

Step 1 : Build a class that implements the javax.servlet.jsp.tagext.Tag interface as follows. Compile it and place it under the web-inf/classes directory (in the appropriate package structure).

```
package examples;

import java.io.*; //// THIS PROGRAM IS EVERY TIME I MEAN WHEN
U REFRESH THAT PARTICULAR
CURRENT DATE THIS CUSTOM TAG WILL DISPLAY

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class ShowDateTag implements Tag {

private PageContext pageContext;
private Tag parent;

public int doStartTag() throws JspException {
return SKIP_BODY;
}

public int doEndTag() throws JspException {
try {
pageContext.getOut().write("" + new java.util.Date());
} catch (IOException ioe) {
throw new JspException(ioe.getMessage());
}

return EVAL_PAGE;
}

public void release() {
}

public void setPageContext(PageContext page) {
this.pageContext = page;
}

public void setParent(Tag tag) {
this.parent = tag;
```

```
}

public Tag getParent() {
return this.parent;
}

}
```

Step 2 : Now we need to describe the tag, so create a file called taglib.tld and place it under the web-inf directory.

```
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">


1.0
1.1
myTag
http://www.mycompany.com/taglib
My own tag library


showDate
examples.ShowDateTag
Show the current date
```

Step 3 : Now we need to tell the web application where to find the custom tags, and how they will be referenced from JSP pages. Edit the web.xml file under the web-inf directory and insert the following XML fragement.

```
http://www.mycompany.com/taglib


/WEB-INF/taglib.tld
```

Step 4 : And finally, create a JSP page that uses the custom tag.

Now restart the server and call up the JSP page! You should notice that every time the page is requested, the current date is displayed in the

browser.

Whilst this doesn't explain what all the various parts of the tag are for (e.g. the tag description, page context, etc) it should get you going. If you use the tutorial (above) and this example, you should be able to grasp what's going on!

5. What are the implicit objects in JSP & differences between them

There are nine implicit objects in JSP.

1. pageContext
2. session
3. request
4. response
5. exception
6. out
7. application
8. config
9. page

These are used for different purposes and actually u no need to create these objects in JSP. JSP container will create these objects automatically.

You can directly use these objects.

Example:

If i want to put my username in the session in JSP.

JSP Page:

In the about page, i am using session object. But this session object is not declared in JSP file, because, this is implicit object and it will be created by the jsp container.

If u see the java file for this jsp page in the work folder of apache tomcat, u will find these objects are created.

15. Is there a way I can set the inactivity lease period on a per-session basis?

Typically, a default inactivity lease period for all sessions is set within your JSPengine admin screen or associated properties file. However, if your JSP engine supports the Servlet 2.1 API, you can manage the inactivity lease period on a per-session basis. This is done by invoking the HttpSession.setMaxInactiveInterval() method, right after the session has been created. For example:

```
<%
session.setMaxInactiveInterval(300);
%>
```

would reset the inactivity period for this session to 5 minutes. The inactivity interval is set in seconds.

16. How can I set a cookie and delete a cookie from within a JSP page?

A cookie, mycookie, can be deleted using the following scriptlet:

```
<%
//creating a cookie
Cookie mycookie = new Cookie("aName","aValue");
response.addCookie(mycookie);
```

```
//delete a cookie
Cookie killMyCookie = new Cookie("mycookie", null);
killMyCookie.setMaxAge(0);
killMyCookie.setPath("/");
response.addCookie(killMyCookie);
%>
```

17. How can I declare methods within my JSP page?

You can declare methods for use within your JSP page as declarations. The methods can then be invoked within any other methods you declare, or within JSP scriptlets and expressions.

Do note that you do not have direct access to any of the JSP implicit objects like request, response, session and so forth from within JSP methods. However, you should be able to pass any of the implicit JSP variables as parameters to the methods you declare. For example:

```
<%!
public String whereFrom(HttpServletRequest req) {
HttpSession ses = req.getSession();
...
return req.getRemoteHost();
}
%>
<%
out.print("Hi there, I see that you are coming in from ");
%>
<%= whereFrom(request) %>
```

Another Example:

file1.jsp:

```
<%@page contentType="text/html"%>
<%!
public void test(JspWriter writer) throws IOException{
writer.println("Hello!");
}
%>
```

file2.jsp

```
<%@include file="file1.jsp"%>
<html>
<body>
<%test(out);% >
</body>
</html>
```

18. How can I enable session tracking for JSP pages if the browser has disabled cookies?

We know that session tracking uses cookies by default to associate a session identifier with a unique user. If the browser does not support cookies, or if cookies are disabled, you can still enable session tracking using URL rewriting. URL rewriting essentially includes the session ID within the link itself as a name/value pair. However, for this to be effective, you need to append the session ID for each and

every link that is part of your servlet response. Adding the session ID to a link is greatly simplified by means of of a couple of methods: response.encodeURL() associates a session ID with a given URL, and if you are using redirection, response.encodeRedirectURL() can be used by giving the redirected URL as input. Both encodeURL() and encodeRedirectedURL() first determine whether cookies are supported by the browser; if so, the input URL is returned unchanged since the session ID will be persisted as a cookie.
Consider the following example, in which two JSP files, say hello1.jsp and hello2.jsp, interact with each other. Basically, we create a new session within hello1.jsp and place an object within this session. The user can then traverse to hello2.jsp by clicking on the link present within the page.Within hello2.jsp, we simply extract the object that was earlier placed in the session and display its contents. Notice that we invoke the encodeURL() within hello1.jsp on the link used to invoke hello2.jsp; if cookies are disabled, the session ID is automatically appended to the URL, allowing hello2.jsp to still retrieve the session object. Try this example first with cookies enabled. Then disable cookie support, restart the brower, and try again. Each time you should see the maintenance of the session across pages. Do note that to get this example to work with cookies disabled at the browser, your JSP engine has to support URL rewriting.
hello1.jsp

```
<%@ page session="true" %>
<%
Integer num = new Integer(100);
session.putValue("num",num);
String url =response.encodeURL("hello2.jsp");
%>
<a href='<%=url%>'>hello2.jsp</a>
```

hello2.jsp

```
<%@ page session="true" %>
<%
Integer i= (Integer )session.getValue("num");
out.println("Num value in session is "+i.intValue());
```

19. How do I use a scriptlet to initialize a newly instantiated bean?
A jsp:useBean action may optionally have a body. If the body is specified, its contents will be automatically invoked when the specified bean is instantiated. Typically, the body will contain scriptlets or jsp:setProperty tags to initialize the newly instantiated bean, although you are not restricted to using those alone. The following example shows the "today" property of the Foo bean initialized to the current date when it is instantiated. Note that here, we make use of a JSP expression within the jsp:setProperty action.

```
<jsp:useBean id="foo" >
<jsp:setProperty name="foo" property="today"
value="<%=java.text.DateFormat.getDateInstance().format(new java.util.Date())
%>"/ >
<%-- scriptlets calling bean setter methods go here --%>
</jsp:useBean >
```

20. How does JSP handle run-time exceptions?
You can use the errorPage attribute of the page directive to have uncaught runtime exceptions

automatically forwarded to an error processing page. For example:
<%@ page errorPage="error.jsp" %>
redirects the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing. Within error.jsp, if you indicate that it is an error-processing page, via the directive:
<%@ page isErrorPage="true" %>
the Throwable object describing the exception may be accessed within the error page via the exception implicit object.
Note: You must always use a relative URL as the value for the errorPage attribute.

21. How do I prevent the output of my JSP or Servlet pages from being cached by the browser?
You will need to set the appropriate HTTP header attributes to prevent the dynamic content output by the JSP page from being cached by the browser. Just execute the following scriptlet at the beginning of your JSP pages to prevent them from being cached at the browser. You need both the statements to take care of some of the older browser versions.

```
<%
response.setHeader("Cache-Control","no-store"); //HTTP 1.1
response.setHeader("Pragma","no-cache"); //HTTP 1.0
response.setDateHeader ("Expires", 0); //prevents caching at the proxy server
%>
```

22. How do I use comments within a JSP page?
You can use "JSP-style" comments to selectively block out code while debugging or simply to comment your scriptlets. JSP comments are not visible at the client.
For example:

```
<%-- the scriptlet is now commented out
<%
out.println("Hello World");
%>
--%>
```

You can also use HTML-style comments anywhere within your JSP page. These comments are visible at the client. For example:
<!-- (c) 2004 javagalaxy.com -->
Of course, you can also use comments supported by your JSP scripting language within your scriptlets. For example, assuming Java is the scripting language, you can have:

```
<%
//some comment
/**
yet another comment
**/
%>
```

23. Can I stop JSP execution while in the midst of processing a request?
Yes. Preemptive termination of request processing on an error condition is a good way to maximize the throughput of a high-volume JSP engine. The trick (asuming Java is your scripting language) is to use the return statement when you want to terminate further processing. For example, consider:

```
<% if (request.getParameter("foo") != null) {
```

```
// generate some html or update bean property
} else {
/* output some error message or provide redirection back to the input form after creating a memento bean updated with the 'valid' form elements that were input. This bean can now be used by the previous form to initialize the input elements that were valid then, return from the body of the _jspService() method to terminate further processing */
return;
}
%>
```

24. Is there a way to reference the "this" variable within a JSP page?
Yes, there is. Under JSP 1.0, the page implicit object is equivalent to "this", and returns a reference to the servlet generated by the JSP page.

25. How do I perform browser redirection from a JSP page?
You can use the response implicit object to redirect the browser to a different resource, as:
response.sendRedirect("http://www.exforsys.com/path/error.html");
You can also physically alter the Location HTTP header attribute, as shown below:

```
<%
response.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
String newLocn = "/newpath/index.html";
response.setHeader("Location",newLocn);
%>
```

You can also use the: <jsp:forward page="/newpage.jsp" /> Also note that you can only use this before any output has been sent to the client. I beleve this is the case with the response.sendRedirect() method as well. If you want to pass any paramateres then you can pass using <jsp:forward page="/servlet/login"> <jsp:param name="username" value="jsmith" /> </jsp:forward>>

26. How do I include static files within a JSP page?]
Answer Static resources should always be included using the JSP include directive. This way, the inclusion is performed just once during the translation phase. The following example shows the syntax:

```
<%@ include file="copyright.html" %>
```

Do note that you should always supply a relative URL for the file attribute. Although you can also include static resources using the action, this is not advisable as the inclusion is then performed for each and every request.

27. What JSP lifecycle methods can I override?
You cannot override the _jspService() method within a JSP page. You can however, override the jspInit() and jspDestroy() methods within a JSP page. jspInit() can be useful for allocating resources like database connections, network connections, and so forth for the JSP page. It is good programming practice to free any allocated resources within jspDestroy().
The jspInit() and jspDestroy() methods are each executed just once during the lifecycle of a JSP page and are typically declared as JSP declarations:

```
<%!
public void jspInit() {
. . .
```

```
}
%>
<%!
public void jspDestroy() {
. . .
}
%>
```

28. Can a JSP page process HTML FORM data?
Yes. However, unlike servlets, you are not required to implement HTTP-protocol specific methods like doGet() or doPost() within your JSP page. You can obtain the data for the FORM input elements via the request implicit object within a scriptlet or expression as:

```
<%
String item = request.getParameter("item");
int howMany = new Integer(request.getParameter("units")).intValue();
%>
```

or

```
<%= request.getParameter("item") %>
```

29. How do I mix JSP and SSI #include?
If you're just including raw HTML, use the #include directive as usual inside your .jsp file.
<!--#include file="data.inc"-->
But it's a little trickier if you want the server to evaluate any JSP code that's inside the included file. If your data.inc file contains jsp code you will have to use <%@ vinclude="data.inc" %> The <!--#include file="data.inc"--> is used for including non-JSP files.

30. How can I implement a thread-safe JSP page?
You can make your JSPs thread-safe by having them implement the SingleThreadModel interface. This is done by adding the directive <%@ page isThreadSafe="false" % > within your JSP page.

31. How do I include static files within a JSP page?
Static resources should always be included using the JSP include directive. This way, the inclusion is performed just once during the translation phase. The following example shows the syntax: Do note that you should always supply a relative URL for the file attribute. Although you can also include static resources using the action, this is not advisable as the inclusion is then performed for each and every request.

32. How do you prevent the Creation of a Session in a JSP Page and why?
By default, a JSP page will automatically create a session for the request if one does not exist. However, sessions consume resources and if it is not necessary to maintain a session, one should not be created. For example, a marketing campaign may suggest the reader visit a web page for more information. If it is anticipated that a lot of traffic will hit that page, you may want to optimize the load on the machine by not creating useless sessions.

33. What is the page directive is used to prevent a JSP page from automatically creating a session:

<%@ page session="false">

34. Is it possible to share an HttpSession between a JSP and EJB? What happens when I change a value in the HttpSession from inside an EJB?
You can pass the HttpSession as parameter to an EJB method, only if all objects in session are serializable.This has to be consider as "passed-by-value", that means that it's read-only in the EJB. If anything is altered from inside the EJB, it won't be reflected back to the HttpSession of the Servlet Container.The "pass-byreference" can be used between EJBs Remote Interfaces, as they are remote references. While it IS possible to pass an HttpSession as a parameter to an EJB object, it is considered to be "bad practice (1)" in terms of object oriented design. This is because you are creating an unnecessary coupling between back-end objects (ejbs) and front-end objects (HttpSession). Create a higher-level of abstraction for your ejb's api. Rather than passing the whole, fat, HttpSession (which carries with it a bunch of http semantics), create a class that acts as a value object (or structure) that holds all the data you need to pass back and forth between front-end/back-end. Consider the case where your ejb needs to support a non-http-based client. This higher level of abstraction will be flexible enough to support it. (1) Core J2EE design patterns (2001)

35. Can a JSP page instantiate a serialized bean?
No problem! The useBean action specifies the beanName attribute, which can be used for indicating a serialized bean. For example:
<jsp:useBean id="shop" type="shopping.CD" beanName="CD" />
<jsp:getProperty name="shop" property="album" />
A couple of important points to note. Although you would have to name your serialized file "filename.ser", you only indicate "filename" as the value for the beanName attribute. Also, you will have to place your serialized file within the WEB-INFjspbeans directory for it to be located by the JSP engine.

36. Can you make use of a ServletOutputStream object from within a JSP page?
No. You are supposed to make use of only a JSPWriter object (given to you in the form of the implicit object out) for replying to clients. A JSPWriter can be viewed as a buffered version of the stream object returned by response.getWriter(), although from an implementational perspective, it is not. A page author can always disable the default buffering for any page using a page directive as:
<%@ page buffer="none" %>

37. Can we implements interface or extends class in JSP?
Yes u can do tht using <%@ page extend="package.className">

38. How can my JSP communicate with java class file
Through
Ex:

39. What are the steps required in adding a JSP Tag Libraries?
Create a TLD file and configure the required class Information.
Create the Java Implementation Source extending the JSP Tag Lib Class (TagSupport).
Compile and package it as loosed class file or as a jar under lib folder in Web Archive File for Class loading.

Place the TLD file under the WEB-INF folder.
Add reference to the tag library in the web.xml file.