## [**JSP interview questions**](http://questions)

## **What is JSP?-** JSP is a dynamic scripting capability for web pages that allows Java as well as a few special tags to be embedded into a web file (HTML/XML, etc). The suffix traditionally ends with .jsp to indicate to the web server that the file is a JSP files. JSP is a server side technology - you can’t do any client side validation with it. The advantages are: a) The JSP assists in making the HTML more functional. b) Platform independence c) Creation of database-driven Web applicationsd) Server-side programming capabilities

**What are the implicit objects?** - Implicit objects are objects that are created by the web container and contain information related to a particular application. Request, response, pageContext, session, application, out, config, page, exception.   
**Is JSP technology extensible?** - Yes. JSP technology is extensible through the development of custom actions, or tags, which are encapsulated in tag libraries.   
**How can I implement a thread-safe JSP page? What are the advantages and Disadvantages of using it?** - 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. With this, instead of a single instance of the servlet generated for your JSP page loaded in memory, you will have N instances of the servlet loaded and initialized, with the service method of each instance effectively synchronized. You can typically control the number of instances (N) that are instantiated for all servlets implementing SingleThreadModel through the admin screen for your JSP engine. More importantly, avoid using the tag for variables. If you do use this tag, then you should set **isThreadSafe to true**, as mentioned above. Otherwise, all requests to that page will access those variables, causing a nasty race condition. SingleThreadModel is not recommended for normal use. There are many pitfalls, including the example above of not being able to use <%! %>. You should try really hard to make them thread-safe the old fashioned way: by making them thread-safe   
**How does JSP handle run-time exceptions?** - You can use the **errorPage** attribute of the page directive to have caught run-time 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" %>** Throwable object describing the exception may be accessed within the error page via the exception implicit object.   
**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 %>   
**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 -- %>  
You can also use **HTML-style** comments anywhere within your JSP page. These comments are visible at the client. For example:  
 <!-- (c) 2004 -->  
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

\*\*/

%>

**Response has already been committed error. What does it mean?** - This error show only when you try to redirect a page after you already have written something in your page. This happens because HTTP specification force the header to be set up before the lay out of the page can be shown (to make sure of how it should be displayed, content-type=”text/html” or “text/xml” or “plain-text” or “image/jpg”, etc.) When you try to send a redirect status (Number is line\_status\_402), your HTTP server cannot send it right now if it hasn’t finished to set up the header. If not starter to set up the header, there are no problems, but if it ’s already begin to set up the header, then your HTTP server expects these headers to be finished setting up and it cannot be the case if the stream of the page is not over… It s simply impossible due to the specification of HTTP 1.0 and 1.1

**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.

**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 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.

**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. 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.   
**Is there a way I can set the inactivity lease period on a per-session basis?** –  
**<% session.setMaxInactiveInterval (300);%>**would reset the inactivity period for this session to 5 minutes. The inactivity interval is set in seconds.  
**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:   
<%  
 Cookie mycookie = new Cookie ("aName","aValue");   
 response.addCookie (mycookie);  
 Cookie killMyCookie = new Cookie ("mycookie", null); killMyCookie.setMaxAge (0);  
 killMyCookie.setPath ("/");  
 response.addCookie (killMyCookie); %>

**How can I prevent the word "null" from appearing in my HTML input text fields when I populate them with a resultset that has null values?** - You could make a simple wrapper function, like   
<%! String blanknull (String s) { return (s == null)? "" : s; } %>  
then use it inside your JSP form, like  
<input type="text" name="shoesize" value="<%=blanknull(shoesize)% >" >  
How can I get to print the stacktrace for an exception occuring within my JSP page?: <%@ page isErrorPage="true" %>  
 <% out.println(" ");  
 PrintWriter pw = response.getWriter();  
 exception.printStackTrace(pw);  
 out.println(" "); %>  
**How do you pass an InitParameter to a JSP?** - The JspPage interface defines the jspInit () and jspDestroy () method which the page writer can use in their pages and are invoked in much the same manner as the init () and destory () methods of a servlet. The example page below enumerates through all the parameters and prints them to the console.   
 <%@ page import="java.util.\*" %>  
 <%! ServletConfig cfg =null;  
 public void jspInit (){  
 ServletConfig cfg=getServletConfig();  
 Enumeration e=cfg.getInitParameterNames ();   
 e.hasMoreElements() {  
 String name=(String)e.nextElement();  
 String value = cfg.getInitParameter(name); System.out.println(name+"="+value); } } %>

.   
 **What are the lifecycle phases of a JSP?**  
JSP is a servlet. When presented with JSP page the JSP engine does the following 7 phases.   
Page translation: -page is parsed, and a java file which is a servlet is created.   
Page compilation: page is compiled into a class file   
Page loading: This class file is loaded.   
Create an instance: - Instance of servlet is created   
\_jspInit () method is called   
\_jspService is called to handle service calls   
\_jspDestroy is called to destroy it when the servlet is not required.   
**What is a translation unit?**   
JSP page can include the contents of other HTML pages or other JSP files. This is done by using the **include directive**. When the JSP engine is presented with such a JSP page it is converted to one servlet class and this is called a translation unit, Things to remember in a translation unit is that page directives effect the whole unit, one variable declaration cannot occur in the same unit more than once, the standard action jsp:useBean cannot declare the same bean twice in one unit.   
**How is JSP used in the MVC model?** JSP is usually used for presentation in the MVC pattern (Model View Controller) i.e. it plays the role of the view. The controller deals with calling the model and the business classes which in turn get the data, this data is then presented to the JSP for rendering on to the client.   
**What is a output comment?** A comment that is sent to the client in the viewable page source. The JSP engine handles an output comment as un-interpreted HTML text, returning the comment in the HTML output sent to the client. You can see the comment by viewing the page source from your Web browser.   
**What is a Hidden Comment?** A comment that documents the JSP page but is not sent to the client. The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags. A hidden comment is not sent to the client, either in the displayed JSP page or the HTML page source. The hidden comment is useful when you want to hide or “comment out” part of your JSP page.   
**What’s the difference between forward and sendRedirect?**When you invoke a **forward request**, the request is sent to another resource on the server, without the client being informed that a different resource is going to process the request. This process occurs completely with in the web container And then returns to the calling method.   
When a **sendRedirect** method is invoked, it causes the web container to return to the browser indicating that a new URL should be requested. Because the browser issues a completely new request any object that are stored as request attributes before the redirect occurs will be lost. This extra round trip a redirect is slower than forward.

**What is difference between custom JSP tags and beans?**

* 1. Custom tags can manipulate JSP content; beans cannot.
  2. Complex operations can be reduced to a significantly simpler form with custom tags than with beans.
  3. Custom tags require quite a bit more work to set up than do beans.
  4. Custom tags usually define relatively self-contained behavior, whereas beans are often defined in one servlet and used in a different servlet or JSP page.
  5. Custom tags are available only in JSP 1.1 and later, but beans can be used in all JSP 1.x versions.

**What are JSP scripting elements?**- JSP scripting elements lets to insert Java code into the servlet that will be generated from the current JSP page. There are three forms: a) Expressions of the form **<%= expression %>** that are evaluated and inserted into the output, b) **Scriptlets o**f the form that are inserted into the servlet’s service method, and c) **Declarations o**f the form <%! Code %1> that are inserted into the body of the servlet class and outside of any existing methods.   
**What are JSP Directives?** A JSP directive affects the overall structure of the servlet class. It usually has the following form :< %@ directive attribute=”value” %> However, you can also combine multiple attribute settings for a single directive, as follows:<%@ directive attribute1=”value1″ attribute 2=”value2″ . . . attribute =”value” %> There are two main types of directive: **page**, which lets to do things like import classes, customize the servlet super class, and the like; and **include**, which lets to insert a file into the servlet class at the time the JSP file is translated into a servlet   
**What are JSP ACTIONS?**- JSP actions are use to constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plug-in. Available actions include: **jsp: include** - Include a file at the time the page is requested. **Jsp: useBean** - Find or instantiate a Java Bean. **Jsp: setProperty** - Set the property of a Java Bean. **Jsp: getProperty** - Insert the property of a Java Bean into the output.  
 **Jsp: forward** - Forward the requests to a new page. **Jsp: plug-in** - Generate browser-specific code that makes an OBJECT or EMBED.  
**How do you pass data (including JavaBeans) to a JSP from a servlet?**-  
 (1) **Request Lifetime:** Using this technique to pass beans, a request dispatcher (using either “include” or forward”) can be called. This bean will disappear after processing this request has been completed.   
 **Servlet: request.** setAttribute (”theBean”, myBean);  
 RequestDispatcher rd = getServletContext (). getRequestDispatcher (”thepage. jsp”); rd. forward (request, response);  
 JSP PAGE :< jsp: useBean id=”theBean” scope=”request” class=”. . . . . ” />

2) **Session Lifetime:** Using this technique to pass beans that are relevant to a particular session over a number of requests. This bean will disappear when the session is invalidated or it times out, or when you remove it.   
Servlet: HttpSession session = request. getSession (true);   
session. putValue (”theBean”, myBean)  
 JSP Page :< jsp: useBean id=”theBean” scope=”session” class=”. . . ” />   
3) **Application Lifetime:** Using this technique to pass beans that are relevant to all servlets and JSP pages in a particular app, for all users. For example, I use this to make a JDBC connection pool object available to the various servlets and JSP pages in my apps. This bean will disappear when the servlet engine is shut down, or when you remove it.   
Servlet: GetServletContext (). setAttribute (”theBean”, myBean);   
JSP PAGE :< jsp: useBean id=”theBean” scope=”application” class=”. . . ” />   
**How can I set a cookie in JSP?**  
 **response. setHeader (”Set-Cookie”, “cookie string”);** To give the response-object to a bean, write a method setResponse (HttpServletResponse response) - to the bean, and in jsp-**file :< % bean. setResponse (response); %>**   
**How can I delete a cookie with JSP?**  
Say that I have a cookie called “foo, ” that I set a while ago & I want it to go away. I simply: <% Cookie killCookie = new Cookie(”foo”, null);   
KillCookie. setPath(”/”); killCookie. setMaxAge (0);   
response. addCookie(killCookie); %>   
**Is there a way to execute a JSP from the comandline or from my own application?** There is a little tool called **JSPExecutor** that allows you to do just that. The developers (Hendrix Schreiber <hs@webapp.de> & Peter Rossbach <pr@webapp.de>) aim was not to write a full blown servlet engine, but to provide means to use JSP for generating source code or reports. Therefore most HTTP-specific features (headers, sessions, etc) are not implemented, i.e. no response line or header is generated. Nevertheless you can use it to precompiled JSP for your website’s  
**What is the differences between directive include and jsp include?  
Ans:** <%@ include>: Used to include static resources during translation time.  
 Used to include dynamic content or static content during runtime.  
**How do I mix JSP and SSI #include?  
Ans:** 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.  
**How to pass information from JSP to included JSP?**Using <%jsp: param> tag.  
**How can my application get to know when an HttpSession is removed**?   Define a Class HttpSessionNotifier which implements HttpSessionBindingListener and implement the functionality what you need in valueUnbound () method.   
Create an instance of that class and put that instance in HttpSession.

**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 () {. . .} %>  
**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.foo.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.   
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>>  
**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" />   
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-INF\jsp\beans directory for it to be located by the JSP engine.  
**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 implementation perspective, it is not. A page author can always disable the default buffering for any page using a page directive as:   
<%@ page buffer="none" %>**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 (assuming 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; } %>  
 **How can I get to view any compilation/parsing errors at the client while developing JSP pages?** With JSWDK 1.0, set the following servlet initialization property within the \WEB-INF\servlets.properties file for your application:   
**jsp.initparams=sendErrToClient=true.** This will cause any compilation/parsing errors to be sent as part of the response to the client   .  
**How do I instantiate a bean whose constructor accepts parameters using the useBean tag**? Consider the following bean: package bar;   
public class FooBean {  
public FooBean (SomeObj arg) {... }//getters and setters here}   
the only way you can instantiate this bean within your JSP page is to use a scriptlet. For example, the following snippet creates the bean with session scope:   
 SomeObj x = new SomeObj (...);  
bar.FooBean foobar = new FooBean(x);   
session.putValue("foobar",foobar); %> You can now access this bean within any other page that is part of the same session .   
**Can I invoke a JSP error page from a servlet?**     Yes, you can invoke the JSP error page and pass the exception object to it from within a servlet. The trick is to create a request dispatcher for the JSP error page, and pass the exception object as a javax.servlet.jsp.jspException request attribute. However, note that you can do this from only within controller servlets. If your servlet opens an OutputStream or PrintWriter, the JSP engine will throw the following translation error:   
java.lang.IllegalStateException: Cannot forward as OutputStream or Writer has already been obtained   
The following code snippet demonstrates the invocation of a JSP error page from within a controller servlet:   
protected void sendErrorRedirect(HttpServletRequest request, HttpServletResponse response, String errorPageURL, Throwable e) throws ServletException, IOException {   
request.setAttribute ("javax.servlet.jsp.jspException", e);   
getServletConfig().getServletContext(). getRequestDispatcher(errorPageURL).forward(request, response); }  
public void doPost(HttpServletRequest request, HttpServletResponse response) {  
try {// do something  
} catch (Exception ex) {  
try {  
sendErrorRedirect(request,response,"/jsp/MyErrorPage.jsp",ex); }  
 catch (Exception e) {  
e.printStackTrace();}}}

**How can you store international / Unicode characters into a cookie?**

One way is, before storing the cookie URLEncode it.   
**URLEnocder.encoder (str**); and use   
**URLDecoder.decode (str)** when you get the stored cookie.

**How can my JSP page communicate with an EJB Session Bean**?      
<%@ page import="javax.naming.\*, javax.rmi.PortableRemoteObject, foo.AccountHome, foo.Account" %>   
<**%!** //declare a "global" reference to an instance of the home interface of the session bean //  
AccountHome accHome=null;   
public void jspInit() { //obtain an instance of the home interface   
InitialContext cntxt = new InitialContext( );   
Object ref= cntxt.lookup("java:comp/env/ejb/AccountEJB");   
accHome = (AccountHome)PortableRemoteObject.narrow(ref,AccountHome.class);}**%>**   
**<%**   
//instantiate the session bean   
Account acct = accHome.create();  
//invoke the remote methods   
acct.doWhatever(...);  
// etc etc...   
**%>**   
[what is the difference between servletcontext and pagecontext?](http://www.geekinterview.com/question_details/37984)  
ServletContext gives the information about the container. PageContext gives the information about the Request.  
There is only one servletcontext for a web application, where as there is only one pagecontext for a page.  
Servlet context is an interface whereas pagecontext is an abstract class.  
A PageContext instance provides access to all the namespaces associated with a JSP page, provides access to several page attributes, as well as a layer above the implementation details. Implicit objects are added the pageContext automatically. Whereas servlet context defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.   
[What is difference between custom tags and JavaBean?](http://www.geekinterview.com/question_details/36521)  
Javabeans are used for encapsulating **data management logic** (i.e., used for storage,

) whereas customtags are used for **computational logic** related to particular request.  
Tags are thread safe, beans are not. It should be made thread safe by developers.  
Tags are aware of the environment in which they execute, beans are not.  
Tags can access implicit objects, bean cannot.

Tags have only page scope. Beans have different scopes.  
Tags are not persistent. Beans are persistent

**How to show the variation in the field if the user is an admin or other users?  
For example, if it is an admin, it should show some colours and for other users it should show some other colour, how is it achieved?**  
1. If you are using MVC architecture, then your Controller class could decide the role based JSP. You may've separate JSPs for each page to be displayed for a each roles.  
2. You could also make use of JSP taglib, to selectively display or block some portion of the JSP code.  
3. One could also use mapping in web.xml to forward the request to separate JSPs based upon the login user.  
[What is the difference between page and pagecontext?](http://www.geekinterview.com/question_details/33862)  
page and pageContext are implicit variables.  
page :   page is of class java.lang.Object,and it refers to instance of generated servlet.It is declared as   
           Object page=this;  
     page cannot be used to directly call the servlet methods.  
1) <%=page.getServletInfo () %>   Error bcoz page is java.lang.Object type.  
 2)   <%=((servlet)page.getServletInfo()%>  <----OK:typecast  
 3)   <%=this.getServletInfo()%> <-------OK  
**pageContext :** pageContext is of type javax.servlet.jsp.PageContext. PageContext class is an abstract class. it do following things;  
 1.provide convenience methods to get and set attributes in diff scopes.  
 2. Provide convenience methods for transferring requests to other resources in the web application.  
    Void include (String relative URL) & void forward (String relative URL)  
  Ex; pageContext.forward ("sn.jsp");  
3.store the references to implicit objects  
**How do you upload a file from JSP?**From Jsp we can upload file like this.............  
You have to give input tag like this in jsp     <**input type="file" name="upload"/>**  
**2)** **<html: form action="/ImportTool.do" enctype="multipart/form-data" onsubmit="return validateFileIfAny()">  
 Please select the file to upload: <html: file property="CSV File" value="" name="browse File" size="20"/>  
In our action form bean, declare a property of org.apache.struts.upload.FormFile.**private FormFile CSVFile;  
/\*\*  
  \* @return Returns the CSV File.  
  \*/  
 public FormFile getCSVFile() {  
  return CSVFile;  
 }  
 /\*\*   \* @param file The CSV File to set.   \*/  
 public void setCSVFile(FormFile file) {  
  CSVFile = file; }

**in u r action class,**   
protected  ActionForward doPerform(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response){  
fileName =  form.getCSVFile();  
// do u r own processing.....}  
[**What is the differene between include directive and include action?**](http://www.geekinterview.com/question_details/32313)  
include directive:< %@ include file="x.jsp"%>  
this is static, means x.jsp combined to with in that jsp when traslating to servlet.  
include action ::<jsp:include page="x.jsp"/>  
we can call this dynamically and included at request processing time.  
[How to generate BAR & PIE Graphs using JSP code. . . ?](http://www.geekinterview.com/question_details/29488) **Use jFreeChart.**  
[How to check the value in the text field is not a number](http://www.geekinterview.com/question_details/28032)?  
onblur=mask(this.form.acno) //acno is field name  
function mask(val){  
  if(isNaN(val.value))  {  
    alert("Not a number");  
    val.focus();  
    val.select();  }  
 [how do we perform redirect action without using response. sendRedirect(" ");](http://www.geekinterview.com/question_details/25212)   
By using Hyper Link Tag In   
HttpServletResponse.getWriter().println("<a href = "Destination url"/>");  
Destination url = "<earname>\<resoursename>"..  
[How to pass java script array to jsp page?](http://www.geekinterview.com/question_details/21927)   
Just create a hidden variable, iterate through the JavaScript array, put the value as some comma separated into the hidden variable and submit to the servlet.   
There is no way u can reference a JSP variable from your JavaScript once the page is processed and the response send.  
[what is difference between getAttribute() and getParameter()?](http://www.geekinterview.com/question_details/18501)  
First one is getAttribute returns object and getParamter returns String.  
Second one is getAttribute is used to call only that are included in the request object and getParamter is used for only form attributes like text box ,drop down box etc and values of url appended.  
**When to use Application Scope?**If the java bean holds nothing specific to the client information and the same java bean has to used while processing a multiple requests we must use application scope.  
**When to use Request Scope?**If multiple jsps involved in processing a request has to use single java bean object while processing a request we must use request scope...  
**When to use the page scope?**When a java Bean has to be used in a page only, during the execution of page we must use page scope.  
**When to use the session scope?**  
If the java bean holds client specific information and the same java bean has to used while processing a multiple requests we must use session scope.  
**What is difference between jsp writer and print writer**?  
The methods of jsp writer are designed to throw IOException when there is a problem. But Printwriter classes are not design to throw an exception even if there is a problem.  
Jsp writer supports buffering but printwriter doesn’t support buffering.

[have developed a web page. i want to restrict the user from seeing my html code from web browser how do i do that?????](http://www.geekinterview.com/question_details/18326)  
**<BODY oncontextmenu=window.event.returnValue=false**> is the syntax for disabling the right click on the html page.  
[How can i restrict the user from Clicking of Back button in any browser](http://www.geekinterview.com/question_details/17932)?   Whenever our page is cached, back button displays. Cached page rather a fresh page. So if we can block the caching. anyhow, the back button will automatically get disabled. We can achieve this working by setting two attributes in META tag. That is   
**("Cache-Control", "no-cache") and ("Pragma", "No-cache");  
2)<script> <head>history. forward ()</head> </script>  
In one class i stored the key and values in the hashmap.so using key value i have to check whether the value for key is correct or not. so how to retrieve the key and values in jsp page?**  
1) You can use it by writing a code in Scriptlets in jsp where u can retrieve the Hashmap (from Session, Request,Application) and pass the key entered by user to check the value.  
or2) use Struts framework and using its logic tag library, (logic itereate and logic: equal) this functionality can be achieved easily.  
[How do you restrict page errors display in the JSP page](http://www.geekinterview.com/question_details/709)?  
1. Whatever errors will be displayed on the jsp on exception should be displayed in a separate page rather than on the same page. This is achieved by putting dis fragment of code in the jsp. This will forward any type of exceptions to the separate page error.jsp.   
In error.jsp define as    
2. The other approach is u define specific exceptions at specific places. Like for all java type exceptions you define in web.xml just after tag as     
java.lang.NumberFormatException   
xxx.jsp    
whenever any NumberFormatException will occur in the jsp then jsp container will search for errorPage attribute of < @page > tag if none is specified then will go to tag of web.xml and will match the exception type with that n display the page specified in location.  
**How do you connect to the database from JSP?**  
To be precise to connect jdbc from jsp is not good idea of course if ur working on dummy projects connecting to msaccess u can very well use the same connection objects and methods in ur scriplets and define ur connection object in init() method.   
   
But if its real time u can use DAO design patterns which is widely used. for ex u write all ur connection object and and sql quires in a defined method later use transfer object [TO ]which is all ur fields have get/set methods and call it in business object[BO] so DAO is accessed with precaution as it is the crucial. Finally u define java bean which is a class holding get/set method implementing serialization thus the bean is called in the jsp. So never connect to jdbc directly from client side since it can be hacked by any one to get ur password or credit card info.

[What are the steps required in adding a JSP Tag Libraries?](http://www.geekinterview.com/question_details/15404)   
1) we have to write the taghandler classes that extends TagSupport   
2)create the jar file by using jar cvf (give one name).jar (packagenane).  
3)create the tld file   
4)place the tld file into the WEB-INF directory  
5)place the jar file into WEB-INF/classes directory  
6)configure the imformation about the tld file in web.xml  
[What are Custom tags. Why do you need Custom tags. How do you create Custom tag](http://www.geekinterview.com/question_details/702)? 1) 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 can use custom tags.   
3)  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).

**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" %>**

**How do I have the JSP-generated servlet subclass my own custom servlet class, instead of the default?**

One should be very careful when having JSP pages extend custom servlet classes as opposed to the default one generated by the JSP engine. In doing so, you may lose out on any advanced optimization that may be provided by the JSPengine. In any case, your new superclass has to fulfill the contract with the JSPngine by: Implementing the HttpJspPage interface, if the protocol used is HTTP, or implementing JspPage otherwise Ensuring that all the methods in the Servlet interface are declared final Additionally, your servlet superclass also needs to do the following:

The service() method has to invoke the \_jspService() method

The init() method has to invoke the jspInit() method

The destroy() method has to invoke jspDestroy()

If any of the above conditions are not satisfied, the JSP engine may throw a translation error. Once the superclass has been developed, you can have your JSP extend it as follows:

<%@ page extends="packageName.ServletName" %<

**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") %>