SERVLETS & JSP INTERVIEW QUESTIONS

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 uninterpreted 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.

JSP Syntax: <!-- comment [<%= expression %>] -->

Example1: <!-- This is a commnet sent to client on <%= (new java.util.Date()).toLocaleString() %> -->

Displays in the page source:

<!-- This is a commnet sent to client on January 24, 2004 -->

What is a Hidden Comment?

A comments 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.

You can use any characters in the body of the comment except the closing --%> combination. If you need to use --%> in your comment, you can escape it by typing --%\>.

JSP Syntax: <%-- comment --%>

Examples:<%@ page language="java" %>

<html>

<head><title>A Hidden Comment </title></head>

<body>

<%-- This comment will not be visible to the colent in the page source --%>

</body>

</html>

What is a Expression?

An expression tag contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file. Because the value of an expression is converted to a String, you can use an expression within text in a JSP file. Like

<%= someexpression %>

<%= (new java.util.Date()).toLocaleString() %>

You cannot use a semicolon to end an expression

What is a Declaration?

A declaration declares one or more variables or methods for use later in the JSP source file. A declaration must contain at least one complete declarative statement. You can declare any number of variables or methods within one declaration tag, as long as they are separated by semicolons. The declaration must be valid in the scripting language used in the JSP file.

```
<%! somedeclarations %>
<%! int i = 0; %>
<%! int a, b, c; %>
```

What is a Scriptlet?

A scriptlet can contain any number of language statements, variable or method declarations, or expressions that are valid in the page scripting language.Within scriptlet tags, you can declare variables or methods to use later in the file (see also Declaration). Write expressions valid in the page scripting language (see also Expression). Use any of the JSP implicit objects or any object declared with a <jsp:useBean> tag. You must write plain text, HTML-encoded text, or other JSP tags outside the scriptlet. Scriptlets are executed at request time, when the JSP engine processes the client request. If the scriptlet produces output, the output is stored in the out object, from which you can display it.

What are implicit objects? List them?

Certain objects that are available for the use in JSP documents without being declared first. These objects are parsed by the JSP engine and inserted into the generated servlet. The implicit objects re listed below

- request
- response
- pageContext
- session
- application
- out
- config
- page
- exception

What is 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 completly with in the web container. When a sendRedirtect 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 completly 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 are the different scope valiues for the <jsp:useBean>?

The different scope values for <jsp:useBean> are

- page
- request
- session
- application

Explain the life-cycle mehtods in JSP.

The generated servlet class for a JSP page implements the HttpJspPage interface of the javax.servlet.jsp package. Hte HttpJspPage interface extends the JspPage interface which inturn extends the Servlet interface of the javax.servlet package. the generated servlet class thus implements all the methods of the these three interfaces. The JspPage interface declares only two mehtods - jspInit() and jspDestroy() that must be implemented by all JSP pages regardless of the client-server protocol. However the JSP specification has provided the HttpJspPage interfaec specifically for the JSp pages serving HTTP requests. This interface declares one method _jspService().

The jspInit()- The container calls the jspInit() to initialize te servlet instance.It is called before any other method, and is called only once for a servlet instance.

The _jspservice()- The container calls the _jspservice() for each request, passing it the request and the response objects.

The jspDestroy()- The container calls this when it decides take the instance out of service. It is the last method called n the servlet instance.

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 does JSP handle run-time exceptions?

You can use the errorPage attribute of the page directive to have uncaught 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. Note: You must always use a relative URL as the value for the errorPage attribute.

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 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" class="com.Bar.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 >
```

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="lastName" value="<%=blanknull(lastName)% >" >
```

What's a better approach for enabling thread-safe servlets and JSPs? SingleThreadModel Interface or Synchronization?

Although the SingleThreadModel technique is easy to use, and works well for low volume sites, it does not scale well. If you anticipate your users to increase in the future, you may be better off implementing explicit synchronization for your shared data. The key however, is to effectively minimize the amount of code that is synchronzied so that you take maximum advantage of multithreading. Also, note that SingleThreadModel is pretty resource intensive from the server\'s perspective. The most serious issue however is when the number of concurrent requests exhaust the servlet instance pool. In that case, all the unserviced requests are queued until something becomes free - which results in poor performance. Since the usage is non-deterministic, it may not help much even if you did add more memory and increased the size of the instance pool.

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());
%>
```

What is the difference b/w variable declared inside a declaration part and variable declared in scriplet part?

Variable declared inside declaration part is treated as a global variable.that means after convertion jsp file into servlet that variable will be in outside of service method or it will be declared as instance variable.And the scope is available to complete jsp and to complete in the converted servlet class.where as if u declare a variable inside a scriplet that variable will be declared inside a service method and the scope is with in the service method.

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 (Hendrik 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 reponseline or header is generated. Nevertheless you can use it to precompile JSP for your website.

What is the difference between JSP and Servlets?

JSP is used mainly for presentation only. A JSP can only be HttpServlet that means the only supported protocol in JSP is HTTP. But a servlet can support any protocol like HTTP, FTP, SMTP etc.

What is difference between custom JSP tags and beans?

Custom JSP tag is a tag you defined. You define how a tag, its attributes and its body are interpreted, and then group your tags into collections called tag libraries that can be used in any number of JSP files. To use custom JSP tags, you need to define three separate components: the tag handler class that defines the tag's behavior ,the tag library descriptor file that maps the XML element names to the tag implementations and the JSP file that uses the tag library.

JavaBeans are Java utility classes you defined. Beans have a standard format for Java classes. Custom tags and beans accomplish the same goals -- encapsulating complex behavior into simple and accessible forms. There are several differences:

Custom tags can manipulate JSP content; beans cannot. Complex operations can be reduced to a significantly simpler form with custom tags than with beans. Custom tags require quite a bit more work to set up than do beans. 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. Custom tags are available only in JSP 1.1 and later, but beans can be used in all JSP 1.x versions.

What are the different ways for session tracking?

Cookies, URL rewriting, HttpSession, Hidden form fields

What mechanisms are used by a Servlet Container to maintain session information?

Cookies, URL rewriting, and HTTPS protocol information are used to maintain session information

What is the difference between GET and POST methods?

In GET your entire form submission can be encapsulated in one URL, like a hyperlink. query length is limited to 255 characters, not secure, faster, quick and easy. The data is submitted as part of URL.

In POST data is submitted inside body of the HTTP request. The data is not visible on the URL and it is more secure.

What is session?

The session is an object used by a servlet to track a user's interaction with a Web application across multiple HTTP requests. The session is stored on the server.

What is servlet mapping?

The servlet mapping defines an association between a URL pattern and a servlet. The mapping is used to map requests to Servlets.

What is servlet context?

The servlet context is an object that contains a information about the Web application and container. Using the context, a servlet can log events, obtain URL references to resources, and set and store attributes that other servlets in the context can use.

What is a servlet?

Servlet is a java program that runs inside a web container.

Can we use the constructor, instead of init(), to initialize servlet?

Yes. But you will not get the servlet specific things from constructor. The original reason for init() was that ancient versions of Java couldn't dynamically invoke constructors with arguments, so there was no way to give the constructor a ServletConfig. That no longer applies, but servlet containers still will only call your no-arg constructor. So you won't have access to a ServletConfig or ServletContext.

How many JSP scripting elements are there and what are they?

There are three scripting language elements: declarations, scriptlets, expressions.

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.

How can I implement a thread-safe JSP page?

You can make your JSPs thread-safe adding the directive <%@ page isThreadSafe="false" % > within your JSP page.

What is the difference in using request.getRequestDispatcher() and context.getRequestDispatcher()?

In request.getRequestDispatcher(path) in order to create it we need to give the relative path of the resource. But in resourcecontext.getRequestDispatcher(path) in order to create it we need to give the absolute path of the resource.

What are the lifecycle of JSP?

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 are context initialization parameters?

Context initialization parameters are specified by the <context-param> in the web.xml file, these are initialization parameter for the whole application.

What is a Expression?

Expressions are act as place holders for language expression, expression is evaluated each time the page is accessed. This will be included in the service method of the generated servlet.

What is a Declaration?

It declares one or more variables or methods for use later in the JSP source file. A declaration must contain at least one complete declarative statement. You can declare any number of variables or methods within one declaration tag, as long as semicolons separate them. The declaration must be valid in the scripting language used in the JSP file. This will be included in the declaration section of the generated servlet.

What is a Scriptlet?

A scriptlet can contain any number of language statements, variable or expressions that are valid in the page scripting language. Within scriptlet tags, you can declare variables to use later in the file, write expressions valid in the page scripting language, use any of the JSP implicit objects or any object declared with a <jsp:useBean>. Generally a scriptlet can contain any java code that are valid inside a normal java method. This will become the part of generated servlet's service method.

Explain the life cycle methods of a Servlet.

The javax.servlet.Servlet interface defines the three methods known as life-cycle method.

- public void init(ServletConfig config) throws ServletException
- public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
- public void destroy()

First the servlet is constructed, then initialized wih the init() method. Any request from client are handled initially by the service() method before delegating to the doXxx() methods in the case of HttpServlet.

The servlet is removed from service, destroyed with the destroy() methid, then garbaged collected and finalized.

What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?

The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.

The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root.

Explain the directory structure of a web application.

The directory structure of a web application consists of two parts.

- A private directory called WEB-INF
- A public resource directory which contains public resource folder.

WEB-INF folder consists of

- web.xml
- classes directory
- lib directory

What are the common mechanisms used for session tracking?

- Cookies
- SSL sessions
- URL- rewriting

Explain ServletContext.

ServletContext interface is a window for a servlet to view it's environment. A servlet can use this interface to get information such as initialization parameters for the web applicationor servlet container's version. Every web application has one and only one ServletContext and is accessible to all active resource of that application.

What is preinitialization of a servlet?

A container doesnot initialize the servlets ass soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

What is the difference between Difference between doGet() and doPost()?

A doGet() method is limited with 2k of data to be sent, and doPost() method doesn't have this limitation. A request string for doGet() looks like the following: http://www.allapplabs.com/svt1?p1=v1&p2=v2&...&pN=vN
doPost() method call doesn't need a long text tail after a servlet name in a request. All parameters are stored in a request itself, not in a request string, and it's impossible to guess the data transmitted to a servlet only looking at a request string.

What is the difference between HttpServlet and GenericServlet?

A GenericServlet has a service() method aimed to handle requests. HttpServlet extends GenericServlet and adds support for doGet(), doPost(), doHead() methods (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() methods (HTTP 1.1). Both these classes are abstract.

What is the difference between ServletContext and ServletConfig?

ServletContext: 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.The ServletContext object is contained within

the ServletConfig object, which the Web server provides the servlet when the servlet is initialized.

ServletConfig: The object created after a servlet is instantiated and its default constructor is read. It is created to pass initialization information to the servlet.

What are the implicit objects?

Implicit objects are objects that are created by the web container and contain information related to a particular request, page, or application. They are: 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 <%! DECLARE %>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 uncaught 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. Note: You must always use a relative URL as the value for the errorPage attribute.

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
<%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 -->

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 commited 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… In this last case it's like you have a file started with <HTML Tag><Some Headers><Body>some output (like testing your variables.) Before you indicate that the file is over (and before the size of the page can be setted up in the header), you try to send a redirect status. 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, 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" class="com.Bar.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 >
```

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());
```

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>
```

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 JSP engine 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.

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);
%>
```

How does a servlet communicate with a JSP page?

The following code snippet shows how a servlet instantiates a bean and initializes it with FORM data posted by a browser. The bean is then placed into the request, and the call is then forwarded to the JSP page, Bean1.jsp, by means of a request dispatcher for downstream processing.

```
public void doPost (HttpServletRequest request, HttpServletResponse response) {
try {
        govi.FormBean f = new govi.FormBean();
        String id = request.getParameter("id");
        f.setName(request.getParameter("name"));
        f.setAddr(request.getParameter("addr"));
        f.setAge(request.getParameter("age"));
        //use the id to compute
        //additional bean properties like info
        //maybe perform a db query, etc.
        // . . .
        f.setPersonalizationInfo(info);
        request.setAttribute("fBean",f);
        getServletConfig().getServletContext().getRequestDispatcher
         ("/jsp/Bean1.jsp").forward(request, response);
        } catch (Exception ex) {
. . .
```

```
  }
}
```

The JSP page Bean1.jsp can then process fBean, after first extracting it from the default request scope via the useBean action.

```
jsp:useBean id="fBean" class="govi.FormBean" scope="request"
/ jsp:getProperty name="fBean" property="name"
/ jsp:getProperty name="fBean" property="addr"
/ jsp:getProperty name="fBean" property="age"
/ jsp:getProperty name="fBean" property="personalizationInfo" /
```

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 JSP engine. In any case, your new superclass has to fulfill the contract with the JSP engine 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" %>
```

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?

By printing out the exception's stack trace, you can usually diagonse a problem better when debugging JSP pages. By looking at a stack trace, a programmer should be able to discern which method threw the exception and which method called that method. However, you cannot print the stacktrace using the JSP out implicit variable, which is of type JspWriter. You will have to use a PrintWriter object instead. The following snippet demonstrates how you can print a stacktrace from within a JSP error 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();
for (Enumeration e=cfg.getInitParameterNames(); e.hasMoreElements();) {
String name=(String)e.nextElement();
String value = cfg.getInitParameter(name);
System.out.println(name+"="+value);
}
}
%>
```

How can my JSP page communicate with an EJB Session Bean?

The following is a code snippet that demonstrates how a JSP page can interact 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 makes J2EE suitable for distributed multitiered Applications?

The J2EE platform uses a multitiered distributed application model. Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multitiered J2EE environment to which the application component belongs. The J2EE application parts are:

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

What is J2EE?

J2EE is an environment for developing and deploying enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, web-based applications.

What are the components of J2EE application?

A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are client components.
- Java Servlet and JavaServer Pages technology components are web components.
- Enterprise JavaBeans components (enterprise beans) are business components.
- Resource adapter components provided by EIS and tool vendors.

What do Enterprise JavaBeans components contain?

Enterprise JavaBeans components contains Business code, which is logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by enterprise beans running in the business tier. All the business code is contained inside an Enterprise Bean which receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage. An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.

Is J2EE application only a web-based?

No, It depends on type of application that client wants. A J2EE application can be web-based or non-web-based. if an application client executes on the client machine, it is a non-web-based J2EE application. The J2EE application can provide a way for users to handle tasks such as J2EE system or application administration. It typically has a graphical user interface created from Swing or AWT APIs, or a command-line interface. When user request, it can open an HTTP connection to establish communication with a servlet running in the web tier.

Are JavaBeans J2EE components?

No. JavaBeans components are not considered J2EE components by the J2EE specification. They are written to manage the data flow between an application client or applet and components running on the J2EE server or between server components

and a database. JavaBeans components written for the J2EE platform have instance variables and get and set methods for accessing the data in the instance variables. JavaBeans components used in this way are typically simple in design and implementation, but should conform to the naming and design conventions outlined in the JavaBeans component architecture.

Is HTML page a web component?

No. Static HTML pages and applets are bundled with web components during application assembly, but are not considered web components by the J2EE specification. Even the server-side utility classes are not considered web components, either.

What can be considered as a web component?

J2EE Web components can be either servlets or JSP pages. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.

What is the container?

Containers are the interface between a component and the low-level platform specific functionality that supports the component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

What are container services?

A container is a runtime support of a system-level entity. Containers provide components with services such as lifecycle management, security, deployment, and threading.

What is the web container?

Servlet and JSP containers are collectively referred to as Web containers. It manages the execution of JSP page and servlet components for J2EE applications. Web components and their container run on the J2EE server.

What is Enterprise JavaBeans (EJB) container?

It manages the execution of enterprise beans for J2EE applications. Enterprise beans and their container run on the J2EE server.

What is Applet container?

IManages the execution of applets. Consists of a Web browser and Java Plugin running on the client together.

How do we package J2EE components?

J2EE components are packaged separately and bundled into a J2EE application for deployment. Each component, its related files such as GIF and HTML files or server-side utility classes, and a deployment descriptor are assembled into a module and added to the J2EE application. A J2EE application is composed of one or more enterprise bean,Web, or application client component modules. The final enterprise solution can use one J2EE application or be made up of two or more J2EE applications, depending on design requirements. A J2EE application and each of its modules has its own deployment descriptor. A deployment descriptor is an XML document with an .xml extension that describes a component's deployment settings.

What is a thin client?

A thin client is a lightweight interface to the application that does not have such operations like query databases, execute complex business rules, or connect to legacy applications.

What are types of J2EE clients?

Following are the types of J2EE clients:

- Applets
- Application clients
- Java Web Start-enabled rich clients, powered by Java Web Start technology
- Wireless clients, based on Mobile Information Device Profile (MIDP) technology

What is deployment descriptor?

A deployment descriptor is an Extensible Markup Language (XML) text-based file with an .xml extension that describes a component's deployment settings. A J2EE application and each of its modules has its own deployment descriptor. For example, an enterprise bean module deployment descriptor declares transaction attributes and security authorizations for an enterprise bean. Because deployment descriptor information is declarative, it can be changed without modifying the bean source code. At run time, the J2EE server reads the deployment descriptor and acts upon the component accordingly.

What is the EAR file?

An EAR file is a standard JAR file with an .ear extension, named from Enterprise ARchive file. A J2EE application with all of its modules is delivered in EAR file.

What is JTA and JTS?

JTA is the abbreviation for the Java Transaction API. JTS is the abbreviation for the Jave Transaction Service. JTA provides a standard interface and allows you to demarcate transactions in a manner that is independent of the transaction manager implementation. The J2EE SDK implements the transaction manager with JTS. But your code doesn't call the JTS methods directly. Instead, it invokes the JTA methods, which then call the lower-level JTS routines. Therefore, JTA is a high level transaction interface that your application uses to control transaction. and JTS is a low level transaction interface and ejb uses behind the scenes (client code doesn't directly interact with JTS. It is based on object transaction service(OTS) which is part of CORBA.

What is JAXP?

JAXP stands for Java API for XML. XML is a language for representing and describing text-based data which can be read and handled by any program or tool that uses XML APIs. It provides standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and create the appropriate JavaBeans component to perform those operations.

What is J2EE Connector?

The J2EE Connector API is used by J2EE tools vendors and system integrators to create resource adapters that support access to enterprise information systems that can be plugged into any J2EE product. Each type of database or EIS has a different resource adapter. Note: A resource adapter is a software component that allows J2EE application components to access and interact with the underlying resource manager. Because a resource adapter is specific to its resource manager, there is typically a different resource adapter for each type of database or enterprise information system.

What is JAAP?

The Java Authentication and Authorization Service (JAAS) provides a way for a J2EE application to authenticate and authorize a specific user or group of users to run it. It is a standard Pluggable Authentication Module (PAM) framework that extends the Java 2 platform security architecture to support user-based authorization.

What is Java Naming and Directory Service?

The JNDI provides naming and directory functionality. It provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a

J2EE application can store and retrieve any type of named Java object. Because JNDI is independent of any specific implementations, applications can use JNDI to access multiple naming and directory services, including existing naming and directory services such as LDAP, NDS, DNS, and NIS.

What is Struts?

A Web page development framework. Struts combines Java Servlets, Java Server Pages, custom tags, and message resources into a unified framework. It is a cooperative, synergistic platform, suitable for development teams, independent developers, and everyone between.

How is the MVC design pattern used in Struts framework?

In the MVC design pattern, application flow is mediated by a central Controller. The Controller delegates requests to an appropriate handler. The handlers are tied to a Model, and each handler acts as an adapter between the request and the Model. The Model represents, or encapsulates, an application's business logic or state. Control is usually then forwarded back through the Controller to the appropriate View. The forwarding can be determined by consulting a set of mappings, usually loaded from a database or configuration file. This provides a loose coupling between the View and Model, which can make an application significantly easier to create and maintain. Controller: Servlet controller which supplied by Struts itself; View: what you can see on the screen, a JSP page and presentation components; Model: System state and a business logic JavaBeans.

What is a servlet?

Servlets are modules that extend request/response-oriented servers,such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database. Servlets are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface.

Whats the advantages using servlets over using CGI?

Servlets provide a way to generate dynamic documents that is both easier to write and faster to run. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension.

What are the general advantages and selling points of Servlets?

A servlet can handle multiple requests concurrently, and synchronize requests. This allows servlets to support systems such as online real-time

conferencing. Servlets can forward requests to other servers and servlets. Thus servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.

Which package provides interfaces and classes for writing servlets?

javax

What's the Servlet Interface?

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.Servlets > Generic Servlet > HttpServlet > MyServlet.

The Servlet interface declares, but does not implement, methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.

When a servlet accepts a call from a client, it receives two objects. What are they?

ServletRequest (which encapsulates the communication from the client to the server) and ServletResponse (which encapsulates the communication from the servlet back to the client). ServletRequest and ServletResponse are interfaces defined inside javax.servlet package.

What information does ServletRequest allow access to?

Information such as the names of the parameters passed in by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that received it. Also the input stream, as ServletInputStream.Servlets use the input stream to get data from clients that use application protocols such as the HTTP POST and GET methods.

What type of constraints can ServletResponse interface set on the client?

It can set the content length and MIME type of the reply. It also provides an output stream, ServletOutputStream and a Writer through which the servlet can send the reply data.

Explain servlet lifecycle?

Each servlet has the same life cycle: first, the server loads and initializes the servlet (init()), then the servlet handles zero or more client requests (service()), after that the server removes the servlet (destroy()). Worth noting that the last step on some servers is done when they shut down.

How does HTTP Servlet handle client requests?

An HTTP Servlet handles client requests through its service method. The service method supports standard HTTP client requests by dispatching each request to a method designed to handle that request.

What is JSP? Describe its concept.

JSP is a technology that combines HTML/XML markup languages and elements of Java programming Language to return dynamic content to the Web client, It is normally used to handle Presentation logic of a web application, although it may have business logic.

What are the lifecycle phases of a JSP?

JSP page looks like a HTML page but 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 affect 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 are context initialization parameters?

Context initialization parameters are specified by the <context-param> in the web.xml file, these are initialization parameter for the whole application and not specific to any servlet or JSP.

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 is a Expression?

Expressions are act as place holders for language expression, expression is evaluated each time the page is accessed.

What is a Declaration?

It declares one or more variables or methods for use later in the JSP source file. A declaration must contain at least one complete declarative statement. You can declare any number of variables or methods within one declaration tag, as long as semicolons separate them. The declaration must be valid in the scripting language used in the JSP file.

What is a Scriptlet?

A scriptlet can contain any number of language statements, variable or method declarations, or expressions that are valid in the page scripting language. Within scriptlet tags, you can declare variables or methods to use later in the file, write expressions valid in the page scripting language, use any of the JSP implicit objects or any object declared with a <jsp:useBean>.

What are the implicit objects? List them.

Certain objects that is available for the use in JSP documents without being declared first. These objects are parsed by the JSP engine and inserted into the generated servlet. The implicit objects are:

- request
- response
- pageContext
- session
- application

- out
- config
- page
- exception

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 are the different scope values for the <jsp:useBean>?

The different scope values for <jsp:useBean> are:

- page
- request
- session
- application

Why are JSP pages the preferred API for creating a web-based client program?

Because no plug-ins or security policy files are needed on the client systems(applet does). Also, JSP pages enable cleaner and more module application design because they provide a way to separate applications programming from web page design. This means personnel involved in web page design do not need to understand Java programming language syntax to do their jobs.

Is JSP technology extensible?

Yes, it is. JSP technology is extensible through the development of custom actions, or tags, which are encapsulated in tag libraries.

What is difference between custom JSP tags and beans?

Custom JSP tag is a tag you defined. You define how a tag, its attributes and its body are interpreted, and then group your tags into collections called tag libraries that can be used in any number of JSP files. Custom tags and beans accomplish the same goals — encapsulating complex behavior into simple and accessible forms. There are several differences:

- Custom tags can manipulate JSP content; beans cannot.
- Complex operations can be reduced to a significantly simpler form with custom tags than with beans.
- Custom tags require quite a bit more work to set up than do beans.
- 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.
- Custom tags are available only in JSP 1.1 and later, but beans can be used in all JSP 1.x version

What are the most common techniques for reusing functionality in object-oriented systems?

The two most common techniques for reusing functionality in object-oriented systems are class inheritance and object composition. Class inheritance lets you define the implementation of one class in terms of another's. Reuse by subclassing is often referred to as white-box reuse. Object composition is an alternative to class inheritance. Here, new functionality is obtained by assembling or composing objects to get more complex functionality. This is known as black-box reuse.

Why would you want to have more than one catch block associated with a single try block in Java?

Since there are many things can go wrong to a single executed statement, we should have more than one catch(s) to catch any errors that might occur.

What language is used by a relational model to describe the structure of a database?

The Data Definition Language

What is JSP? Describe its concept.

JSP is Java Server Pages. The JavaServer Page concept is to provide an HTML document with the ability to plug in content at selected locations in the document.

What does the JSP engine do when presented with a JavaServer Page to process?

The JSP engine builds a servlet. The HTML portions of the JavaServer Page become Strings transmitted to print methods of a PrintWriter object. The JSP tag portions result in calls to methods of the appropriate JavaBean class whose output is translated into more calls to a println method to place the result in the HTML document.

What is servlet?

Servlets are modules that extend request/response-oriented servers, such as java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.

What is the difference between an applet and a servlet?

- Servlets are to servers what applets are to browsers
- Applets must have graphical user interfaces whereas servlets have no graphical user interfaces.

What is the difference between doPost and doGet methods?

- doGet() method is used to get information, while doPost() method is used for posting information
- doGet() requests can't send large amount of information and is limited to 240-255 characters. However, doPost()requests passes all of its data, of unlimited length
- A doGet() request is appended to the request URL in a query string and this allows the exchange is visible to the client, whereas a doPost() request passes directly over the socket connection as part of its HTTP request body and the exchange are invisible to the client.

What is the life cycle of a servlet?

Each Servlet has the same life cycle:

- A server loads and initializes the servlet by init () method
- The servlet handles zero or more client's requests through service() method.
- The server removes the servlet through destroy() method.

Who is loading the init() method of servlet?

Web server

What are the different servers available for developing and deploying Servlets?

- Java Web Server
- JRun
- Apache Server
- Netscape Information Server
- Web Logic

How many ways can we track client and what are they?

The servlet API provides two ways to track client state and they are:

- Using Session tracking
- Using Cookies.

What is session tracking and how do you track a user session in servlets?

Session tracking is a mechanism that servlets use to maintain state about a series requests from the same user across some period of time. The methods used for session tracking are:

- User Authentication: occurs when a web server restricts access to some of its resources to only those clients that log in using a recognized username and password
- Hidden form fields: fields are added to an HTML form that is not displayed in the client's browser. When the form containing the fields is submitted, the fields are sent back to the server
- URL rewriting: every URL that the user clicks on is dynamically modified or rewritten to include extra information. The extra information can be in the form of extra path information, added parameters or some custom, server-specific URL change
- Cookies: a bit of information that is sent by a web server to a browser and which can later be read back from that browser
- HttpSession: places a limit on the number of sessions that can exist in memory. This limit is set in the session. maxresidents property.

What is Server-Side Includes (SSI)?

Server-Side Includes allows embedding servlets within HTML pages using a special servlet tag. In many servlets that support servlets, a page can be processed by the server to include output from servlets at certain points inside the HTML page. This is accomplished using a special internal SSINCLUDE, which processes the servlet tags. SSINCLUDE servlet will be invoked whenever a file with an. shtml extension is requested. So HTML files that include server-side includes must be stored with an . shtml extension.

What are cookies and how will you use them?

Cookies are a mechanism that a servlet uses to have a client hold a small amount of state-information associated with the user.

- Create a cookie with the Cookie constructor: public Cookie(String name, String value)

- A servlet can send a cookie to the client by passing a Cookie object to the addCookie() method of HttpServletResponse: public void HttpServletResponse. addCookie(Cookie cookie)
- A servlet retrieves cookies by calling the getCookies() method of HttpServletRequest: public Cookie[ ] HttpServletRequest. getCookie().

Is it possible to communicate from an applet to servlet and how many ways and how?

Yes, there are three ways to communicate from an applet to servlet and they are:

- HTTP Communication (Text-based and object-based)
- Socket Communication
- RMI Communication

What is connection pooling?

With servlets, opening a database connection is a major bottleneck because we are creating and tearing down a new connection for every page request and the time taken to create connection will be more. Creating a connection pool is an ideal approach for a complicated servlet. With a connection pool, we can duplicate only the resources we need to duplicate rather than the entire servlet. A connection pool can also intelligently manage the size of the pool and make sure each connection remains valid. A number of connection pool packages are currently available. Some like DbConnectionBroker are freely available from Java Exchange Works by creating an object that dispenses connections and connection Ids on request. The ConnectionPool class maintains a Hastable, using Connection objects as keys and Boolean values as stored values. The Boolean value indicates whether a connection is in use or not. A program calls getConnection() method of the ConnectionPool for getting Connection object it can use; it calls returnConnection() to give the connection back to the pool.

Why should we go for interservlet communication?

Servlets running together in the same server communicate with each other in several ways. The three major reasons to use interservlet communication are:

- Direct servlet manipulation - allows to gain access to the other currently loaded servlets and perform certain tasks (through the ServletContext object)
- Servlet reuse - allows the servlet to reuse the public methods of another servlet

- Servlet collaboration - requires to communicate with each other by sharing specific information (through method invocation)

Is it possible to call servlet with parameters in the URL?

Yes. You can call a servlet with parameters in the syntax as (?Param1 = xxx || m2 = yyy).

What is Servlet chaining?

Servlet chaining is a technique in which two or more servlets can cooperate in servicing a single request. In servlet chaining, one servlet's output is piped to the next servlet's input. This process continues until the last servlet is reached. Its output is then sent back to the client.

How do servlets handle multiple simultaneous requests?

The server has multiple threads that are available to handle requests. When a request comes in, it is assigned to a thread, which calls a service method (for example: doGet(), doPost() and service()) of the servlet. For this reason, a single servlet object can have its service methods called by many threads at once.

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:

- The JSP assists in making the HTML more functional. Servlets on the other hand allow outputting of HTML but it is a tedious process.
- It is easy to make a change and then let the JSP capability of the web server you are using deal with compiling it into a servlet and running it.

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:

- Expressions of the form <%= expression %> that are evaluated and inserted into the output
- Scriptlets of the form<% code %>that are inserted into the servlet's service method
- Declarations of the form <%! Code %>that are inserted into the body of the servlet class, 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” . . . attributeN =”valueN” %> There are two main types of directive: page, which lets to do things like import classes, customize the servlet superclass, 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 Predefined variables or implicit objects?

To simplify code in JSP expressions and scriptlets, we can use eight automatically defined variables, sometimes called implicit objects. They are request, response, out, session, application, config, pageContext, and page.

What are JSP ACTIONS?

JSP actions use 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 plugin. Available actions include: jsp:include - Include a file at the time the page is requested. jsp:useBean - Find or instantiate a JavaBean. jsp:setProperty - Set the property of a JavaBean. jsp:getProperty - Insert the property of a JavaBean into the output. jsp:forward - Forward the requester to a newpage. Jsp: plugin - Generate browser-specific code that makes an OBJECT or EMBED

How do you pass data (including JavaBeans) to a JSP from a servlet?

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=”. . . . . ” />

Session Lifetime: Using this technique to pass beans that are relevant to a particular session (such as in individual user login) 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); /* You can do a request dispatcher here, or just let the bean be visible on the next request */ JSP Page:<jsp:useBean id=”theBean” scope=”session” class=”. . . ” />

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); %>

How are Servlets and JSP Pages related?

JSP pages are focused around HTML (or XML) with Java codes and JSP tags inside them. When a web server that has JSP support is asked for a JSP page, it checks to see if it has already compiled the page into a servlet. Thus, JSP pages become servlets and are transformed into pure Java and then compiled, loaded into the server and executed.

Can we implement an interface in a JSP?

No

What is the difference between ServletContext and PageContext?

ServletContext: Gives the information about the container.

PageContext: Gives the information about the Request

What is the difference in using request.getRequestDispatcher() and context.getRequestDispatcher()?

request.getRequestDispatcher(path): In order to create it we need to give the relative path of the resource

context.getRequestDispatcher(path): In order to create it we need to give the absolute path of the resource.

How to pass information from JSP to included JSP?

Using <%jsp:param> tag.

What is the difference between directive include and jsp include?

<%@ include>: Used to include static resources during translation time

JSP include: Used to include dynamic content or static content during runtime.

What is the difference between RequestDispatcher and sendRedirect?

RequestDispatcher: server-side redirect with request and response objects.

sendRedirect: Client-side redirect with new request and response objects.

How does JSP handle runtime exceptions?

Using errorPage attribute of page directive and also we need to specify isErrorPage=true if the current page is intended to URL redirecting of a JSP.

How do you delete a Cookie within a JSP?

```
Cookie mycook = new Cookie("name","value");
response.addCookie(mycook);
Cookie killmycook = new Cookie("mycook","value");
killmycook.setMaxAge(0);
killmycook.setPath("/");
killmycook.addCookie(killmycook);
```

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.

What is the architechture of servlet package?

Servlet Interface is the central abstraction. All servlets implements this Servlet .Interface either direclty or indirectly (may implement or extend Servlet Interfaces sub classes or sub interfaces)

Servlet ⟶ Generic Servlet ⟶ HttpServlet (Class) -- we will extend this class to handle GET / PUT HTTP requests ⟶ MyServlet

What is the difference between HttpServlet and GenericServlet?

A GenericServlet has a service() method to handle requests. HttpServlet extends GenericServlet added new methods:

- doGet()
- doPost()
- doHead()

- doPut()
- doOptions()
- doDelete()
- doTrace() methods

Which code line must be set before any of the lines that use the PrintWriter?

setContentType() method must be set.

Which protocol will be used by browser and servlet to communicate?

HTTP

In how many ways we can track the sessions?

- By URL rewriting
- Using Session object:

Getting Session form HttpServletRequest object

```
HttpSession session = request.getSession(true);
```

Get a Value from the session

```
session.getValue(session.getId());
```

Adding values to session

```
cart = new Cart();
session.putValue(session.getId(), cart);
```

At the end of the session, we can inactivate the session by using the following command

session.invalidate();

- Using cookies
- Using hidden fields

How Can You invoke other web resources (or other servlet / jsp) ?

Servelt can invoke other Web resources in two ways: indirect and direct.

Indirect Way: Servlet will return the resultant HTML to the browser which will point to another Servlet (Web resource)

Direct Way: We can call another Web resource (Servelt / Jsp) from Servelt program itself, by using RequestDispatcher object. You can get this object using getRequestDispatcher("URL") method. You can get this object from either a request or a Context.

Example:

```
RequestDispatcher dispatcher = request.getRequestDispatcher("/jspsample.jsp");
if (dispatcher != null){
    dispatcher.forward(request, response);
```

```
}
```

How Can you include other Resources in the Response?

Using include method of a RequestDispatcher object. Included WebComponent (Servlet / Jsp) cannot set headers or call any method (for example, setCookie) that affects the headers of the response.

Example:

```
RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/banner");
if (dispatcher != null){
        dispatcher.include(request, response);
}
```

What is the difference between the getRequestDispatcher(String path) ServletRequest interface and ServletContext interface?

The getRequestDispatcher(String path) method of ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root. The getRequestDispatcher(String path) method of ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root. If the resource is not available, or if the server has not implemented a RequestDispatcher object for that type of resource, getRequestDispatcher will return null. Your servlet should be prepared to deal with this condition.

What is the use of ServletContext?

Using ServletContext, We can access data from its environment. Servlet context is common to all Servlets so all Servlets share the information through ServeltContext.

Is there any way to generate PDF'S dynamically in servlets?

We need to use iText. A open source library for java. Please refer sourceforge site for sample servlet examples.

What is the difference between using getSession(true) and getSession(false) methods?

getSession(true) - This method will check whether already a session is existing for the user. If a session is existing, it will return that session object, Otherwise it will create new session object and return taht object.

getSession(false) - This method will check existence of session. If session exists, then it returns the reference of that session object, if not, this methods will return null.

Why are JSP pages the preferred API for creating a web-based client program?

Because no plug-ins or security policy files are needed on the client systems(applet does). Also, JSP pages enable cleaner and more module application design because they provide a way to separate applications programming from web page design. This means personnel involved in web page design do not need to understand Java programming language syntax to do their jobs.

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 uncaught 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. Note: You must always use a relative URL as the value for the errorPage attribute.

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 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.

```
value="<%=java.text.DateFormat.getDateInstance().format(new
java.util.Date()) %>"/ >
<%-- scriptlets calling bean setter methods go here --%>"
```

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 JSP engine 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.

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);
%>
```

How does a servlet communicate with a JSP page?

The following code snippet shows how a servlet instantiates a bean and initializes it with FORM data posted by a browser. The bean is then placed into the request, and the call is then forwarded to the JSP page, Bean1.jsp, by means of a request dispatcher for downstream processing.

```
public void doPost (HttpServletRequest request, HttpServletResponse response) {
  try {
      govi.FormBean f = new govi.FormBean();
      String id = request.getParameter("id");
     f.setName(request.getParameter("name"));
      f.setAddr(request.getParameter("addr"));
      f.setAge(request.getParameter("age"));
      //use the id to compute
      //additional bean properties like info
      //maybe perform a db query, etc.
      // . . .
      f.setPersonalizationInfo(info);
      request.setAttribute("fBean",f);
      getServletConfig().getServletContext().getRequestDispatcher
                ("/jsp/Bean1.jsp").forward(request, response);
      } catch (Exception ex) {
  }
}
```

The JSP page Bean1.jsp can then process fBean, after first extracting it from the default request scope via the useBean action.

jsp:useBean id="fBean" class="govi.FormBean" scope="request"

/ jsp:getProperty name="fBean" property="name"

/ jsp:getProperty name="fBean" property="addr"

/ jsp:getProperty name="fBean" property="age"

/ jsp:getProperty name="fBean" property="personalizationInfo" /

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 JSP engine. In any case, your new superclass has to fulfill the contract with the JSP engine 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" %>

How can I get to print the stacktrace for an exception occuring within my JSP page?

By printing out the exception's stack trace, you can usually diagonse a problem better when debugging JSP pages. By looking at a stack trace, a programmer should be able to discern which method threw the exception and which method called that method. However, you cannot print the stacktrace using the JSP out implicit variable, which is of type JspWriter. You will have to use a PrintWriter object instead. The following snippet demonstrates how you can print a stacktrace from within a JSP error 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();
for (Enumeration e=cfg.getInitParameterNames(); e.hasMoreElements();) {
String name=(String)e.nextElement();
String value = cfg.getInitParameter(name);
System.out.println(name+"="+value);
}
}
%>
```

How can my JSP page communicate with an EJB Session Bean?

The following is a code snippet that demonstrates how a JSP page can interact 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...
%>
```

Can we implement an interface in a JSP?

No