www.questpond.com

The Software Interview Question Bank

-- Maintained by Shivprasad Koirala shiv_koirala@yahoo.com

Do you have a question which can cost somebody a good job mail me at shiv_koirala@yahoo.com

Do you have a suggestion / tips and tricks which can make job searcher easier mail me at shiv_koirala@yahoo.com.

How to buy the book

BPB has done a great job of making this book reach to places where i can hardly imagine. But just incase its not near to your place mail bpb@bol.net.in.
If you are from India you can contact one of the shops below:-
MUMBAI-22078296/97/022-22070989
KOLKATA-22826518/19
HYDERABAD-24756967,24756400
BANGALORE-25587923,25584641
AHMEDABAD-26421611
BHATINA(PUNJAB)-2237387,
CHENNAI-28410796,28550491
DELHI/NEW DELHI-23254990/91,23325760,26415092,24691288

Pakistan
M/s. Vanguard Books P Ltd, 45 The Mall, Lahore, Pakistan (Tel: 0092-42-7235767, 7243783 and 7243779 and Fax: 7245097)
E-mail: vbl@brain.net.pk

If you are not from india or pakistan :-
Ray McLennan, director,Motilal (UK) Books of India,367 High Street.
London Colney,
St.Albans, Hertfordshire,AL2 1EA, U.K.
Tel. +44 (0)1727 761 677,Fax.+44 (0)1727 761
357,info@mlbduk.com,www.mlbduk.com

If you want to purchase the book directly through BPB Publication's delhi , India :-

bpb@bol.net or bpb@vsnl.com

Contents

From the Author

First thing thanks to all those who have sent me complaints and also appreciation for what ever titles i have written till today. But interview question series is very near to my heart as i can understand the pain of searching a job. Thanks to my publishers (BPB) , readers and reviewers to always excuse all my stupid things which i always do.

So why is this PDF free ?. Well i always wanted to distribute things for free specially when its a interview question book which can fetch a job for a developer. But i am also bounded with publishers rules and regulations. And why not they have a whole team of editor, printing guys, designers, distributors, shopkeepers and including me. But again the other aspect, readers should know of what they are buying , the quality and is it really useful to buy this book. So here are sample free questions which i am giving out free to the readers to see the worth of the book.

I can be contacted at shiv_koirala@yahoo.com its bit difficult to answer all answers but as i get time i do it.

We have recently started a career counselling drive absolutely free for new comers and experienced guys. So i have enlisted the following guys on the panel. Thanks to all these guys to accept the panel job of consulting. Feel free to shoot them questions just put a title in the mail saying “Question about Career”. I have always turned up to them when i had some serious career decision to take.

Shivprasad Koirala :- Not a great guy but as i have done the complete book i have to take up one of the positions. You can contact me at shiv_koirala@yahoo.com for technical career aspect.

Tapan Das :- If you think you are aiming at becoming a project manager he is the right person to consult. He can answer all your questions regarding how to groom your career as a project manager tapand@vsnl.com.

Kapil Siddharth :- If you are thinking to grow as architect in a company then he is a guy. When it comes to role model as architect i rate this guy at the top. You can contact him at kapilsiddharth@hotmail.com

Second if you think you can help the developers mail me at shiv_koirala@yahoo.com and if i find you fitting in the panel i will display your mail address. Please note there are no financial rewards as such but i am sure you will be proud of the work you are doing and whos knows what can come up.

Lets make Software Industry a better place to work ..... Happy Job Hunting and Best of Luck

Career Path Institute

Author runs the “Softwar Career Path Insitute” personally in mumbai. If you are interested you can contact him regarding admissions at shiv_koirala@yahoo.com. Our courses are mainly targetting from how to get a job perspective.

Below are some of the courses offered :-

-- Interview preparation course two days (Saturday and Sunday Batch). (C# , SQL Server)
-- Full one year course for C# , SQL Server

4.Servlets and JSP

(B) What are Servlets?

Servlets are small program which execute on the web server. They run under web server environment exploiting the functionalities of the web server.

(B) What are advantages of servlets over CGI?

In CGI for every request there is a new process started which is quiet an overhead. In servlets JVM stays running and handles each request using a light weight thread. In CGI if there are 1000 request then 1000 CGI program is loaded in memory while in servlets there are 1000 thread and only one copy of the servlet class.

(B) Can you explain Servlet life cycle?

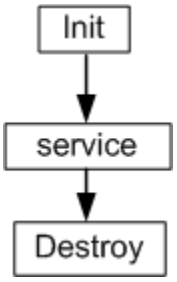

Figure 4.1 : - Servlet Life cycle

There are three methods which are very important in servlet life cycle i.e. "init", "service" and "destroy". Server invokes "init ()" method when servlet is first loaded in to the web server memory. Servlet reads HTTP data provided in HTTP request in the "service ()" method. Once initialized servlet remains in memory to process subsequent request. So for every HTTP request "service ()" method of the servlet is called. Finally when server unloads the "servlet ()" from the memory it calls the "destroy" method which can be used to clean up any resource the servlet is consuming.

(B) What are the two important API's in for Servlets?

Two important packages are required to build servlet "javax.servlet" and "javax.servlet.http". They form the core of Servlet API. Servlets are not part of core Java but are standard extensions provided by Tomcat.

(B) Can you explain in detail “javax.servlet” package?

javax.servlet package has interfaces and classes which define a framework in which servlets can operate. Let’s first make a walk through of all the interfaces and methods and its description.

Interfaces in javax.servlet

Servlet Interface

This interface has the init( ), service( ), and destroy( ) methods that are called by the server during the life cycle of a servlet.

Following are the method in Servlet interface :-

void destroy( ):- Executed when servlet is unloaded from the web server memory.

ServletConfig getServletConfig() :- Returns back a ServletConfig object that contains initialization data.

String getServletInfo( ):- Returns a string describing the servlet.

init method :- Called for first time when the servlet is initialized by the web server.

void service() method :- Called to process a request from a client.

ServletConfig Interface

This interface is implemented by the servlet container. Servlet can access any configuration data when its loaded. The methods declared by this interface are summarized here:

Following are the methods in ServletConfig interface:-

ServletContext getServletContext():- Gives the servlet context.

String getInitParameter(String param):- Returns the value of the initialization parameter named param.

Enumeration getInitParameterNames() :- Returns an enumeration of all initialization parameter names.

String getServletName( ) :- Returns the name of the invoking servlet.

(I)What's the use of ServletContext?

ServletContext Interface

It gives information about the environment. It represents a Servlet's view of the Web Application.Using this interface servlet can access raw input streams to Web Application resources, virtual directory translation, a common mechanism for logging information, and an application scope for binding objects.

Following are the methods defined in ServletContext Interface

Object getAttribute(String attr) :- Returns the value of the server attribute named attr.

String getMimeType(String file) :- Gives MIME type for a file.

String getRealPath(String vpath) :- Gives the actual physical path for a virtual path.

String getServerInfo( ) :- You can get the server information using this function.

void log(String s) :- Used to write to server log.

void log(String s, Throwable e) :- Writes s and the stack trace for e to the servlet log.

void setAttribute(String attr, Object val) :- Sets the attribute specified by attr to the value passed in val.

ServletRequest Interface

The ServletRequest interface is implemented by the servlet container. It gives data regarding client request.

Following are the methods defined in ServletRequest Interface

Object getAttribute(String attr) :- Returns the value of the attribute named attr.

String getCharacterEncoding( ) :- Returns the character encoding of the request.

int getContentLength( ) :- Gives the size of the request. If no size is there then it returns -1.

String getContentType( ) :- Returns the type of the request. A null value is returned if the type cannot be determined.

ServletInputStream getInputStream( ) :- Returns a ServletInputStream that can be used to read binary data from the request.

String getParameter(String pname) :- Returns the value of the parameter named pname.

Enumeration getParameterNames( ) :- Returns an enumeration of the parameter names for this request.

String[ ] getParameterValues(String name) :- Returns an array containing values associated with the parameter specified by name.

String getProtocol( ) :- Gives back protocol description.

BufferedReader getReader( ) :- Returns a buffered reader that can be used to read text from the request.

String getRemoteAddr() :-Returns client IP address.

String getRemoteHost() :- Returns client host name.

String getScheme( ) :- Return what's the transmission protocol HTTP , FTP etc.

String getServerName() :- Returns the name of the server.

int getServerPort() :- Returns the port number.

ServletResponse Interface

The ServletResponse interface is implemented by the servlet containerUsed to give response back to the client.

Following are the methods defined in ServletResponse Interface

String getCharacterEncoding() :- Returns back character encoding.

ServletOutputStream getOutputStream() :- Returns a ServletOutputStream that can be used to write binary data to the response.

PrintWriter getWriter() :- Returns a PrintWriter that can be used to write character data to the response.

void setContentLength(int size) :- Sets the content length for the response to size.

void setContentType(String type) :- Sets the content type for the response to type.

GenericServlet Class

The GenericServlet class provides implementations of the basic life cycle methods for a servlet. GenericServlet implements the Servlet and ServletConfig interfaces.

void log(String s)

void log(String s, Throwable e)

Here, s is the string to be appended to the log, and e is an exception that occurred.

Now let's revise through different classes.

ServletInputStream Class

This class extends InputStream. It is implemented by the servlet container and provides an input stream that a servlet developer can use to read the data from a client request. It defines the default constructor. In addition, a method is provided to read bytes from the stream.

int readLine(byte[ ] buffer, int offset, int size) :- Here, buffer is the array into which size bytes are placed starting at offset. The method returns the actual number of bytes read or –1 if an end-of-stream condition is encountered.

ServletOutputStream Class

The ServletOutputStream class extends OutputStream. It is implemented by the servlet container and provides an output stream that a servlet developer can use to write data to a client response. A default constructor is defined. It also defines the "print()" and "println()" methods, which output data to the stream.

Servlet Exception Classes

javax.servlet defines two exceptions. The first is ServletException, which indicates that a servlet problem has occurred. The second is unavailableException, which extends ServletException. It indicates that a servlet is unavailable.

(I) How do we define an application level scope for servlet?

(B) What's the difference between GenericServlet and HttpServlet?

HttpServlet class extends GenericServlet class which is an abstract class to provide HTTP protocol-specific functionalities. Most of the java application developers extend HttpServlet class as it provides more HTTP protocol-specific functionalities. You can see in HttpServlet class doGet (), doPOst () methods which are more targeted towards HTTP protocol specific functionalities. For instance we can inherit from GenericServlet class to make something like MobileServlet. So GenericServlet class should be used when we want to write protocol specific implementation which is not available. But when we know we are making an internet application where HTTP is the major protocol its better to use HttpServlet.

(B) Can you explain in detail javax.servlet.http package?

The javax.servlet.http package inherits from "javax.servlet" package and supplies HTTP protocol specific functionalities for JAVA developers. If you are aiming at developing HTTP application you will find "javax.servlet.HTTP" more comfortable than "javax.servlet".

So let's revisit through the interfaces and methods

HttpServletRequest Interface

Below are the lists of methods in HttpServletRequest Interface:-

String getAuthType( ):- Returns the type of authentication.

Cookie[ ] getCookies( ) :- Returns the collection of cookies for the request.

long getDateHeader(String field) :- Returns the value of the date header field named field.

String getHeader(String field) :- Returns the value of the header field named field.

Enumeration getHeaderNames( ) :- Returns an enumeration of the header names.

int getIntHeader(String field) :- Returns the int equivalent of the header field named field.

String getMethod( ) :- What type of method does this request have POST , GET etc.

String getPathInfo( ) :- Returns any path information that is located after the servlet path and before a query string of the URL.

String getPathTranslated( ) :- Returns any path information that is located after the servlet path and before a query string of the URL after translating it to a real path.

String getQueryString( ) :- Returns any query string in the URL.

String getRemoteUser( ) :- Returns the name of the user who issued this request.

String getRequestedSessionId( ) :- Returns the ID of the session.

String getRequestURI( ) :- Returns the URI.

StringBuffer getRequestURL( ) :- Returns the URL.

String getServletPath( ) :- Returns that part of the URL that identifies the servlet.

HttpSession getSession( ) :- Returns the session for this request. If a session does not exist, one is created and then returned.

HttpSession getSession(boolean new) :- If new is true and no session exists, creates and returns a session for this request. Otherwise, returns the existing session for this request. This section is explained in more detail in further questions.

boolean isRequestedSessionIdFromCookie( ) :- Returns true if the cookie has the session id.

boolean isRequestedSessionIdFromURL( ) :- Gives true if the URL has session id.

boolean isRequestedSessionIdValid( ) :- Return true if the session is valid in the current context.

HttpServletResponse Interface

The HttpServletResponse interface is implemented by the servlet container. It enables a servlet to formulate an HTTP response to a client. Several constants are defined. These correspond to the different status codes that can be assigned to an HTTP response.

Below are the methods and functions for the interface

void addCookie(Cookie cookie) :-Adds cookie to the HTTP response.

String encodeURL(String url) :- Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of URL. Otherwise, returns URL. All URLs generated by a servlet should be processed by this method.

String encodeRedirectURL(String url) :- Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of URL. Otherwise, returns URL. All URLs passed to sendRedirect( ) should be processed by this method.

void sendError(int c) :- Sends the error code c to the client.

void sendError(int c, String s) :- Sends the error code c and message s to the client.

void sendRedirect(String url) :- Redirects the client to url.

void setDateHeader(String field, long msec) :- Adds field to the header with date value equal to msec (milliseconds since midnight, January 1, 1970, GMT).

void setHeader(String field, String value) :- Adds field to the header with value equal to value.

void setIntHeader(String field, int value) :- Adds field to the header with value equal to value.

void setStatus(int code) :- Sets the status code for this response to code.

HttpSession Interface

HTTP protocol is a stateless protocol and this interface enables to maintain sessions between requests.

Object getAttribute(String attr) :- Returns the value associated with the name passed in attr. Returns null if attr is not found.

Enumeration getAttributeNames( ) :- Returns an enumeration of the attribute names associated with the session.

long getCreationTime( ) :- Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when this session was created.

String getId( ) :- Returns the session ID.

long getLastAccessedTime( ) :- Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the client last made a request for this session.

void invalidate() :- Invalidates this session and removes it from the context.

boolean isNew( ) :- Returns true if the server created the session and it has not yet been accessed by the client.

void removeAttribute(String attr) :- Removes the attribute specified by attr from the session.

void setAttribute(String attr, Object val) :- Associates the value passed in val with the attribute name passed in attr.

HttpSessionBindingListener

The HttpSessionBindingListener interface is implemented by objects that need to be notified when they are bound to or unbound from an HTTP session. The methods that are invoked when an object is bound or unbound are

void valueBound(HttpSessionBindingEvent e)

void valueUnbound(HttpSessionBindingEvent e)

Here, e is the event object that describes the binding.

Cookie Class

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. For example, assume that a user visits an online store. A cookie can save the user's name, address, and other information. The user does not need to enter this data each time he or she visits the store.

A servlet can write a cookie to a user's machine via the addCookie( ) method of the HttpServletResponse interface. The data for that cookie is then included in the header of the HTTP response that is sent to the browser.

The names and values of cookies are stored on the user's machine. Some of the information that is saved for each cookie includes name of the cookie, value of the cookie, expiration date of the cookie and domain/path of the cookie.

The expiration date determines when this cookie is deleted from the user's machine. If an expiration date is not explicitly assigned to a cookie, it is deleted when the current browser session ends. Otherwise, the cookie is saved in a file on the user's machine.

The domain and path of the cookie determine when it is included in the header of an HTTP request. If the user enters a URL whose domain and path match these values, the cookie is then supplied to the Web server. Otherwise, it is not. There is one constructor for Cookie. It has the signature shown here:

Cookie(String name, String value) :- Here, the name and value of the cookie are supplied as arguments to the constructor. The methods of the Cookie class are

Object clone( ) :- Returns a copy of this object.

String getComment( ) :- Returns the comment.

String getDomain( ) :- Returns the domain.

int getMaxAge( ) :- Returns the maximum age (in seconds).

String getName( ) :- Returns the name.

String getPath( ) :- Returns the path.

boolean getSecure( ) :- Returns true if the cookie is secure. Otherwise, returns false.

String getValue( ) :- Returns the value.

int getVersion( ) :- Returns the version.

void setComment(String c) :- Sets the comment to c.

void setDomain(String d) :- Sets the domain to d.

void setMaxAge(int secs) :- Sets the maximum age of the cookie to secs. This is the number of seconds after which the cookie is deleted.

void setPath(String p) :- Sets the path to p.

void setSecure(boolean secure) :- Sets the security flag to secure.

void setValue(String v) :- Sets the value to v.

void setVersion(int v) :- Sets the version to v.

HttpServlet Class

The HttpServlet class extends GenericServlet. It is commonly used when developing servlets that receive and process HTTP requests.

void doDelete(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP DELETE.

void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP GET.

void doOptions(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP OPTIONS.

void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP POST.

void doPut(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP PUT.

void doTrace(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Handles an HTTP TRACE.

long getLastModified(HttpServletRequest req) :- Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified.

void service(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException :- Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and response, respectively.

HttpSessionEvent Class

HttpSessionEvent encapsulates session events. It extends EventObject and is generated when a change occurs to the session.

HttpSession getSession( )

It returns the session in which the event occurred.

The HttpSessionBindingEvent Class

The HttpSessionBindingEvent class extends HttpSessionEvent. It is generated when a listener is bound to or unbound from a value in an HttpSession object. It is also generated when an attribute is bound or unbound. Here are its constructors:

HttpSessionBindingEvent(HttpSession session, String name)

HttpSessionBindingEvent(HttpSession session, String name, Object val)

Here, session is the source of the event, and name is the name associated with the object that is being bound or unbound. If an attribute is being bound or unbound, its value is passed in Val.

String getName( ) :- The getName( ) method obtains the name that is being bound or unbound. Its constructor is shown here:

HttpSession getSession( ) :- The getSession( ) method, shown next, obtains the session to which the listener is being bound or unbound:

Object getValue( ) :- The getValue( ) method obtains the value of the attribute that is being bound or unbound. It is shown here:

(B) What's the architecture of a Servlet package?

In the previous questions we saw all the servlet packages. But the basic architecture of the servlet packages is as shown below.

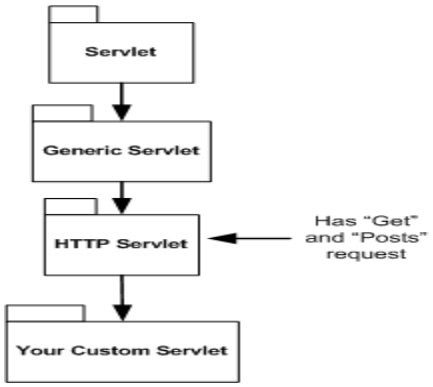

Figure 4.2 : - Servlet package in action

At the top of all is the main servlet interface which is implemented by the generic servlet. But generic servlet does not provide implementation specific to any protocol. HTTP servlet further inherits from the generic servlet and provides HTTP implementation like "Get" and "Post". Finally comes our custom servlet which inherits from HTTP Servlet.

(B) Why is HTTP protocol called as a stateless protocol?

A protocol is stateless if it can remember difference between one client request and the other. HTTP is a stateless protocol because each request is executed independently without any knowledge of the requests that came before it.

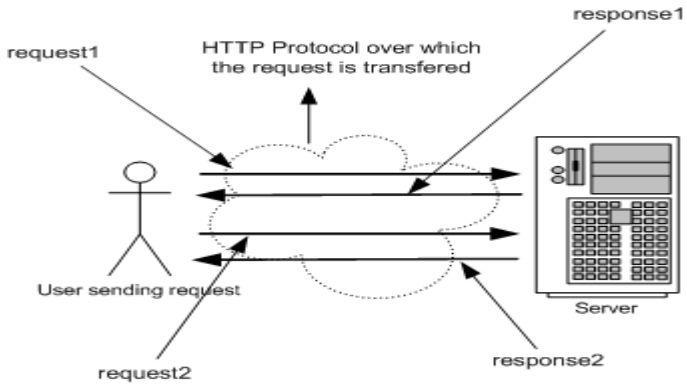

Figure 4.3 : - HTTP Protocol in action

Above is a pictorial presentation of how a stateless protocol operates. User first sends "request1" and server responds with "response1". When the same user comes back with "request2" server treats this as new user and has no idea that it's the same user who has come with the request. In short every request is a new request for the HTTP protocol so it's called as a stateless protocol.

(B) What are the different ways we can maintain state between requests?

Following are the different ways of maintaining state's between stateless requests:-

- √ URL rewriting
- √ Cookies
- √ Hidden fields
- √ Sessions

(B) What is URL rewriting?

It's based on the concept of attaching a unique ID (which is generated by the server) in the URL of response from the server. So the server attaches this unique ID in each URL.

When the client sends a requests it sends back this ID with the request also which helps the server to identify the client uniquely.

Note: - A sample code for URL rewriting is in the CD in "URLRewriting" folder.

Below is a snippet which is extracted from the code which is provided in the CD.

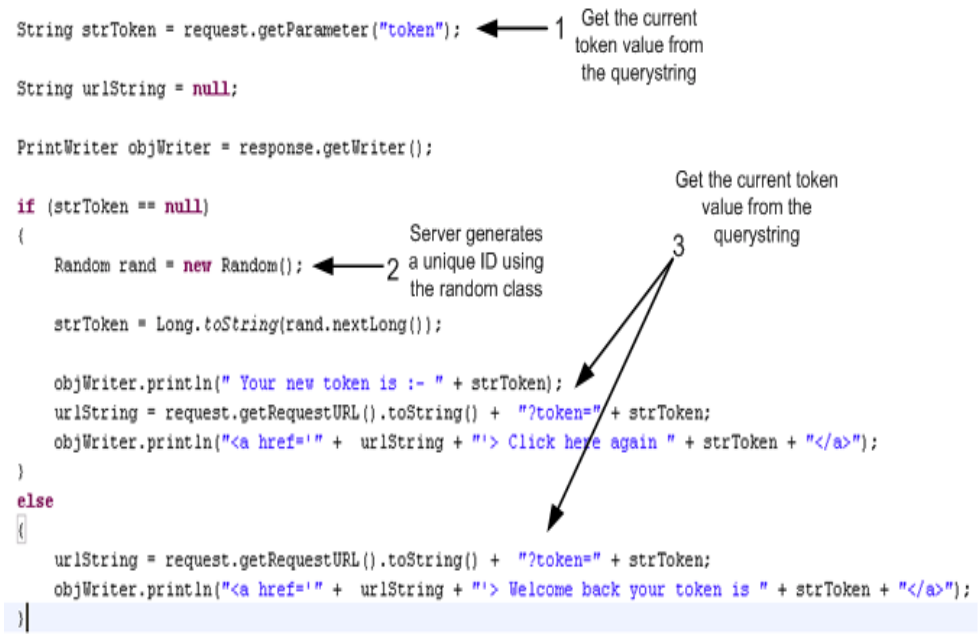

Figure 4.4 : - URLRewriting in Action

In this sample we have generated a unique ID using the random class of java. This unique ID is then sent in the query string (see step 3 of the above snippet) back to the client. When the client comes back to the server the server first gets the token value from the query string and thus identifying the client uniquely.

(B) What are cookies?

Cookies are piece of data which are stored at the client's browser to track information regarding the user usage and habits. Servlets sends cookies to the browser client using the HTTP response headers. When client gets the cookie information it's stored by the browser

in the client's hard disk. Similarly client returns the cookie information using the HTTP request headers.

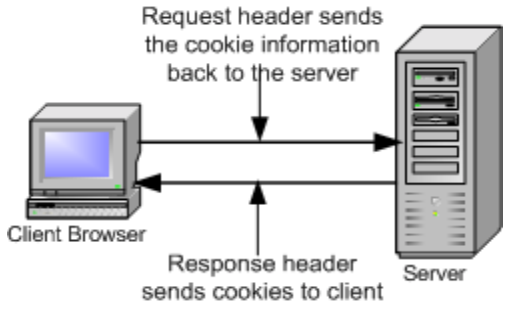

Figure 4.5 : - Cookies in action

```
// get the cookies value in string variable
String strfavoritebook =  request.getParameter("BooksPreference");
// create a new cookie with the value passed
Cookie favoritecookiebook =  new Cookie("favoritecookiebook", strfavoritebook);
// Add a simple comment
favoritecookiebook.setComment("User prefers this");
// add the cookies to the response object
response.addCookie(favoritecookiebook);
// by default there us nothing selected
String result = "still not selected";
if (request.getCookies()!= null)
{
Cookie[] cookies = request.getCookies();
// Loop through all the cookies and get the cookies value
for(int i = 0;i < cookies.length;i++)
{
  if(cookies[i].getName().equals("favoritecookiebook"))
  {
    result = cookies[i].getValue();
    break;
  }}}
// Finally using printwriter object to write the cookies back to the browse
PrintWriter out =  response.getWriter();
out.println("" + "");
// and display the out put
out.println("<H2> So your favourite book is " + result  + " </H2>");
out.println("");
```

1 Creates a new cookie

2 Adds the cookie to the response headers

3 Gets all the cookies in the current request

4 Get the current cookie value

Figure 4.6 : - Cookies code walkthrough

Above is a simple snippet which shows how to use cookies. To create a cookie you need to use the "Cookie" class. In the above snippet step1 creates a cookie using the "Cookie" class. In this case "favoritecookiebook" is the name of the cookie. In Step 2 you can see how a cookie has been added to the response. Step 3 code shows how to get all the cookies which has come in the current request header. Step 4 shows the way to get a cookie from a collection in this case we wanted to retrieve "favoritecookiebook" from the cookies collection.

Note: - You can get the above code in "cookies" folder.

(B)What are sessions in Servlets?

(B)What's the difference between getSession(true) and getSession(false) ?

Session's can be considered as a series of related interactions between client and the server that take place over a period of time. Because HTTP is a stateless protocol these series of interactions are difficult to track. That's where we can use HttpSession object to save in between of these interactions so that server can co-relate between interactions between clients.

```
Step 1 ──► HttpSession presentSession = request.getSession(true);
Step 2 ──► String strName = (String)presentSession.getAttribute("Name");
           PrintWriter objWriter = response.getWriter();
           if (strName == null)
           {
               objWriter.println(" Inserted first time in to session");
               String strobj = new String("Data from Session");
Step 3 ──► presentSession.setAttribute("Name", strobj);
               strName = strobj;
           }
```

Figure 4.7 : - Session code walkthrough

Above is the code snippet which displays session data. Step1 returns an HttpSession object from the request object. “true” parameter in the “getsession” function ensures that we get a session object if there is no current session object in request which ensures that we never get a null session object. In Step 2 we are using the “getattribute” function to return the session value. In step 3 we are setting the session value with “Name” key.

> *Note: - In source code folder we have a sample project “sessions” which can make your concept clearer.*

(I)What’s the difference between “doPost” and “doGet” methods?

When we invoke a servlet the servlet engine passes information to service () method of the servlet. Service method evaluates whether it’s a GET or a POST request type. GET and POST differ from the way data is passed to the server. In GET data is passed as a query string attached to the URL for instance http://www.xyz.com/x.asp?Name=Shiv. So when you pass data through get method you can see the data in your browser URL which can be bad thing for critical data like password. In POST method data is passed through normal input stream. You will not be able to view data like how we get for GET method.

So for POST method below is the HTML code.

```
<form name='MyServlet' method='POST' >
```

For GET method below is the HTML code

```
<form name='MyServlet' method='GET'>
```

(I)Which are the different ways you can communicate between servlets?

Below are the different ways of communicating between servlets:-

- √ Using RequestDispatcher object.
- √ Sharing resource using “ServletContext ()” object.
- √ Include response of the resource in the response of the servlet.
- √ Calling public methods of the resource.
- √ Servlet chaining.

(I)What is functionality of “RequestDispatcher” object?

```
public class clsMaths extends HttpServlet
{
    public void service(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
    {
    RequestDispatcher dispatch;
    String strWhattoDo = request.getParameter("Whattodo");

    if (strWhattoDo == "A")
    {
    dispatch =  getServletContext().getRequestDispatcher("/servlet/clsAdd");
    }
    else
    {
    dispatch =  getServletContext().getRequestDispatcher("/servlet/clsMultiply");
    }
    if (dispatch == null)
    {
    response.sendError(response.SC_NO_CONTENT);
    return;
    }
    HttpSession session = request.getSession();
    dispatch.forward(request, response);
```

Figure 4.8 : - RequestDispatcher in action

Above is the code snippet for “requestdispatcher”. Code snippet “getServletContext().getRequestDispatcher("/servlet/clsAdd"); “gets the “requestdisptacherobject”. In order to call the other servlet class we call the forward method which in turn invokes the service method of the other servlets.

> *Note: - There is a sample project “requestdispatcher” which you can open and see. In order to see how the dispatcher will work you can pass the “WhattoDo” as either “A” or “M” as shown in the figure below. Corresponding servlets are called depending on this value.*

Figure 4.9 : - Running the dispatcher project through URL

(I) How do we share data using "getServletContext ()"?

Using the "getServletContext ()" we can make data available at application level. So servlets which lie in the same application can get the data. Below are important snippets which you will need to add, get and remove data which can be shared across servlets.

```
Step1 ──► getServletContext().setAttribute("Customer",Customer.instance());
Step2 ──► Customer objCustomer = (Customer)getServletContext().getAttribute("Customer");
Step3 ──► getServletContext().removeAttribute("Customer");
```

Figure 4.10 : - Sharing data using "getServletContext ()"

You can see in Step1 how "getServletContext().setAttribute()" method can be used add new objects to the "getServletContext" collection. Step2 shows how we can get the object back using "getAttribute()" and manipulate the same. Step3 shows how to remove an already shared application object from the collection. "getServletContext()" object is shared across servlets with in a application and thus can be used to share data between servlets.

(B) Explain the concept of SSI?

Server Side Includes (SSI) are commands and directives placed in Web pages that are evaluated by the Web server when the Web page is being served. SSI are not supported by all web servers. So before using SSI read the web server documentation for the support. SSI is useful when you want a small part of the page to be dynamically generated rather than loading the whole page again.

Below is the code for SSI which needs to be inserted in between the HTML tags.

<SERVLET CODE=MyServlet CODEBASE=path

initparam1=initvalue1 initparam2=initvalue2>

<PARAM NAME=param1 VALUE=value1>

<PARAM NAME=param2 VALUE=value2>

</SERVLET>

Here the CODE attribute specifies the servlet name. The CODEBASE attribute indicates the servlet location. If you are using a servlet deployed in the same Web server, you can omit the CODEBASE attribute. You can pass any request parameters to the servlet using the PARAM tags.

Below is the code how the SSI looks in the HTML

<HTML>

<HEAD>

<TITLE>Using SSL</TITLE>

<BODY>

<SERVLET CODE=MyServlet CODEBASE=path

initparam1=initvalue1 initparam2=initvalue2>

<PARAM NAME=param1 VALUE=value1>

<PARAM NAME=param2 VALUE=value2>

</SERVLET>

</BODY>

</HTML>

(I) What are filters in JAVA?

Filters are nothing but simple java classes which can manipulate request before it reaches the resource on the web server. Resource can be a HTML file, servlet class, JSP etc. It can also intercept responses sent back to client and thus can manipulate the response before they reach the client browser.

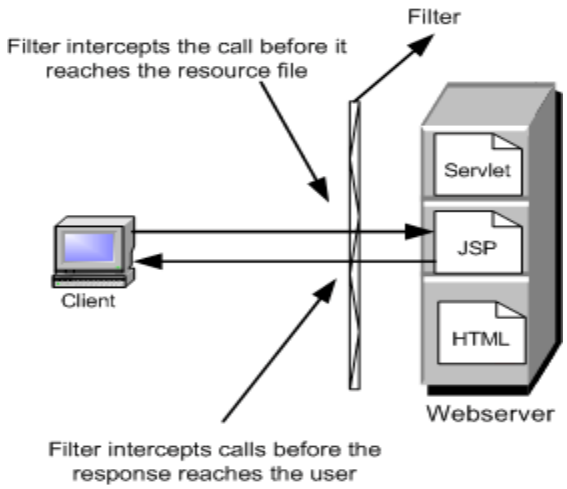

Figure 4.11 : - Filter in action

(A)Can you explain in short how do you go about implementing filters using Apache Tomcat?

Twist: - Explain step by step of how to implement filters?

In order to make a class as a filter we need to inherit from the “Filter” class. You can see the same in the code sample below in Step1.

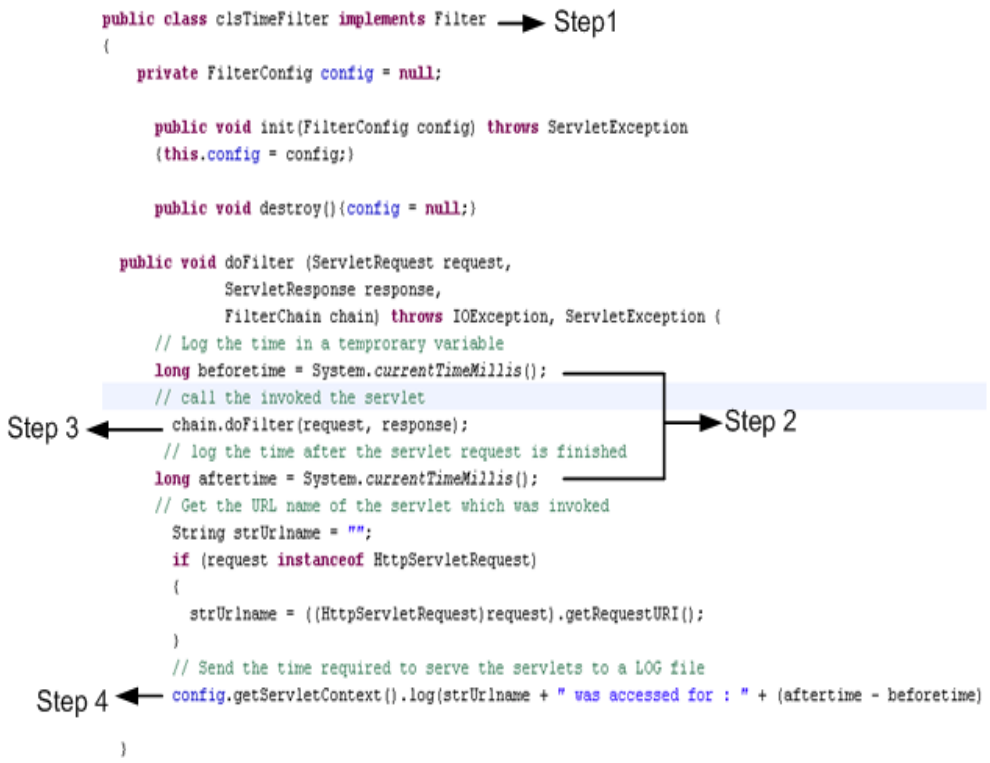

Figure 4.12 : - Filter code walkthrough

Above class “clsTimeFilter” logs the time required for resource to load. You can see in the above code in Step2 we have taken beforetime and aftertime values. In between of them we have called the “dofilter” (Which is Step 3). Finally at Step4 we send the difference of the aftertime and beforetime to the log file.

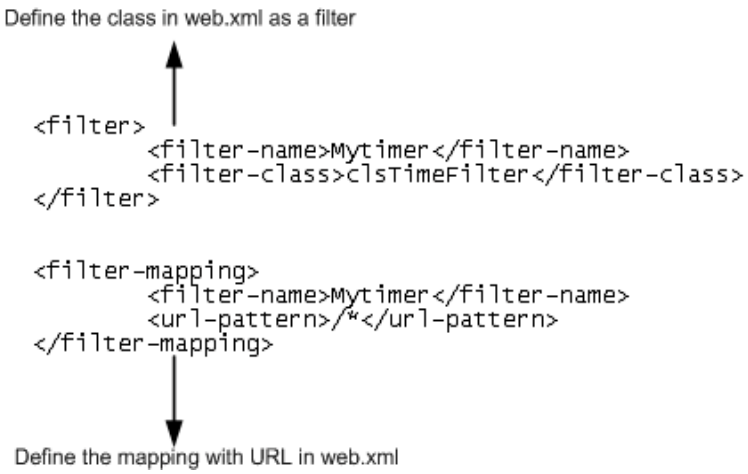

Figure 4.13 : - Filter mapped to URL

Once we compile the class file we need to do two important steps one is define the filter and map the filter to the URL. As we are using tomcat we need to define the same in "web.xml" file. Above is a snippet of "web.xml" file with both the entries. The first entry defines the filter name and second maps it to a URL. Currently the URL pattern is "*" that means any servlet requested will invoke the "clsTimeFilter" filter.

Once you have deployed the filter and you request a URL you can see the output sent by the filter to the LOG of tomcat.

```
May 15, 2006 3:10:49 PM org.apache.catalina.core.ApplicationContext log
INFO: /servlets-examples/servlet/HelloWorldExample was accessed for : 15Milli Seconds
```

Figure 4.14 : - Log file entry

Note:-You can get the source code for the filter in "Filters" folder.

Note: - Running JSP is a breeze if the web server supports it. In CD we have provided Tomcat 5.5 installation. Tomcat 5.5 support JSP. You only need to create a folder for your project in webapps folder and put in your JSP files.

(I) what's the difference between Authentication and authorization?

Authentication is the process the application identifies that you are who. For example when a user logs into an application with a username and password, application checks that the entered credentials against its user data store and responds with success or failure. Authorization, on the other hand, is when the application checks to see if you're allowed to do something. For instance are you allowed to do delete or modify a resource.

(B) Explain in brief the directory structure of a web application?

Below is the directory structure of a web application:-

webapp/

WEB-INF/web.xml

WEB-INF/classes

WEB-INF/lib

- √ The webapp directory contains the JSP files, images, and HTML files. The webapp directory can also contain subdirectories such as images or html or can be organized by function, such as public or private.
- √ The WEB-INF/web.xml file is called the deployment descriptor for the Web application. This file contains configuration information for the Web application, including the mappings of URLs to servlets and filters. The web.xml file also contains configuration information for security, MIME type mapping, error pages, and locale settings
- √ The WEB-INF/classes directory contains the class files for the servlets, JSP files, tag libraries, and any other utility classes that are used in the Web application.
- √ The WEB-INF/lib directory contains JAR files for libraries that are used by the Web application. These are generally third-party libraries or classes for any tag libraries used by the Web application.

(B) Can you explain JSP page life cycle?

When first time a JSP page is request necessary servlet code is generated and loaded in the servlet container. Now until the JSP page is not changed the compiled servlet code serves any request which comes from the browser. When you again change the JSP page the JSP engine again compiles a servlet code for the same.

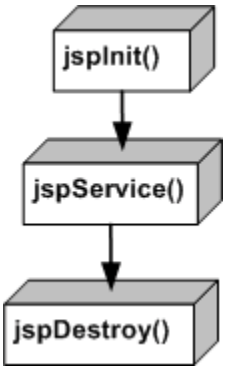

Figure 4.15 : - JSP page life cycle

√ JSP page is first initialized by jspInit() method. This initializes the JSP in much the same way as servlets are initialized, when the first request is intercepted and just after translation.

√ Every time a request comes to the JSP, the container generated _jspService() method is invoked, the request is processed, and response generated.

√ When the JSP is destroyed by the server, the jspDestroy() method is called and this can be used for clean up purposes.

(B) What is EL?

EL stands for expression language. An expression language makes it possible to easily access application data.In the below expression amountofwine variable value will be rendered.

```
<b>There are ${amountofwine} litres of wine in the bottle. </b>
```

(I) how does EL search for an attribute?

EL parser searches the attribute in following order:

√ Page

√ Request

√ Session (if it exists)

√ Application

If no match is found for then it displays empty string.

(B) What are the implicit EL objects in JSP?

Following are the implicit EL objects:-

PageContext: The context for the JSP page.

Provides access to various objects for instance:-

servletContext: The context for the JSP page's servlet and any web components contained in the same application.

session: The session object for the client.

request: The request triggering the execution of the JSP page.

response: The response returned by the JSP page. See Constructing Responses.

In addition, several implicit objects are available that allow easy access to the following objects:

param: Maps a request parameter name to a single value

paramValues: Maps a request parameter name to an array of values

header: Maps a request header name to a single value

headerValues: Maps a request header name to an array of values

cookie: Maps a cookie name to a single cookie

initParam: Maps a context initialization parameter name to a single value

Finally, there are objects that allow access to the various scoped variables described in Using Scope Objects.

pageScope: Maps page-scoped variable names to their values

requestScope: Maps request-scoped variable names to their values

sessionScope: Maps session-scoped variable names to their values

applicationScope: Maps application-scoped variable names to their values

For instance the below snippet will indentify the browser used by the client.

```
<b>Browser:</b> ${header["user-agent"]}<br/>
```

(I) How can we disable EL?

You can disable using isELIgnored attribute of the page directive:

```
<%@ page isELIgnored ="true|false" %>
```

(I) what is JSTL?

JSTL is also called as JSP tag libraries. They are collection of custom actions which can be accessed as JSP tags.

Note: - In CD we have Jakarta tag libs jakarta-taglibs-standard-1.1.2.zip which you can use to practice sample tags and see how it works. When you unzip the file there are two files jstl.jar and standard.jar which you need to copy it in to your application's WEB-INF/lib directory.

(A) Can you explain in short what the different types of JSTL tags are?

Tags are classified in to four groups:-

- √ Core tags
- √ Formatting tags
- √ XML tags
- √ SQL tags

Core tags

<c:if> for conditional flow

<c:forEach> and <c:forTokens> for iteration

<c:choose>...<c:when>....<c:otherwise> for selective flow between mutually exclusive code

<c:set> and <c:remove> for working with scoped variables

<c:out> for rendering the value of variables and expressions

<c:catch> for working with Java exceptions

<c:url> for creating and working with URLs

Formatting tags

Used to format and display text, the date, the time, and numbers. Below are some frequently used tags:-

<fmt:formatNumber>: To render numerical value with specific precision or format

<fmt:formatDate>: To render date and time values in a specific format (and according to international locale-specific conventions)

<fmt:message>: To display an internationalized message (for example, a message in a different language using a different character set)

XML tags

XML tags are meant to process XML data. It supports data-parsing, transforming XML, plus data and flow control based on XPath expressions. These tags are used only when you need to work directly, within the JSP, with XML data.

SQL tags

They are designed to work directly with SQL tags. But most of the time you will see they are used for prototyping and not for final product.

Just before we move ahead with some other questions. Let's how do we install JSTL so that everything works. Below are three installation steps:-

√ Unzip "jakarta-taglibs-standard-1.1.2.zip" and you will see two files jstl.jar and standard.jar in lib directory.

√ Copy both jstl.jar and standard.jar to "\webapps\ROOT\WEB-INF\lib" directory in tomcat.

√ Copy all tld files from the unzipped location to \webapps\ROOT\WEB-INF" directory.

√ Modify the web.xml file to include all TLD files. Below is snippet of some tld's included in web.xml file.

```
<taglib>
    <taglib-uri>http://java.sun.com/jstl/fmt</taglib-uri>
    <taglib-location>/WEB-INF/fmt.tld</taglib-location>
  </taglib>
  <taglib>
    <taglib-uri>http://java.sun.com/jstl/fmt-rt</taglib-uri>
    <taglib-location>/WEB-INF/fmt-rt.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/core</taglib-uri>
    <taglib-location>/WEB-INF/c.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/core-rt</taglib-uri>
    <taglib-location>/WEB-INF/c-rt.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/sql</taglib-uri>
    <taglib-location>/WEB-INF/sql.tld</taglib-location>
  </taglib>

  <taglib>
```

```
    <taglib-uri>http://java.sun.com/jstl/sql-rt</taglib-uri>
    <taglib-location>/WEB-INF/sql-rt.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/x</taglib-uri>
    <taglib-location>/WEB-INF/x.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/x-rt</taglib-uri>
    <taglib-location>/WEB-INF/x-rt.tld</taglib-location>
  </taglib>
```

(I) How can we use beans in JSP?

JSP provides three tags to work with beans:-

√ <jsp:useBean id="bean name" class="bean class" scope = "page | request | session |application "/>

Bean name = the name that refers to the bean.

Bean class = name of the java class that defines the bean.

√ <jsp:setProperty name = "id" property = "someProperty" value = "someValue" />

id = the name of the bean as specified in the useBean tag.

property = name of the property to be passed to the bean.

value = value of that particular property .

√ <jsp:getProperty name = "id" property = "someProperty" />

Here the property is the name of the property whose value is to be obtained from the bean.Below is a code snippet which shows how MyUserClass is used and the values accessed.

```
<jsp:useBean id="user" class="MyUserClass" scope="session"/>
<HTML>
<BODY>
You entered<BR>
Name: <%= user.getUsername() %><BR>
Email: <%= user.getEmail() %><BR>
</BODY>
</HTML>
```

(I) What is the use of <jsp:include> ?

It includes the output of one JSP in to other JSP file at the location of the tag.

Below is the syntax for the same:-

```
<jsp:include page="...url.." flush="true or false"/>
```

page: A URL that is relative to the current JSP page at request time

flush: Determines if the buffer for the output is flushed immediately, before the included page's output.

(I) What is <jsp:forward> tag for ?

It forwards the current request to another JSP page. Below is the syntax for the same:-

```
<jsp:forward page="...url..." />
```

We can also forward parameter to the other page using the param tag

```
<jsp:forward page="..url...">
  <jsp:param ..../>
</jsp:forward>
```

(B) What are JSP directives?

JSP directives do not produce any output. They are used to set global values like class declaration, content type etc. Directives have scope for entire JSP file. They start with <%@ and ends with %>. There are three main directives that can be used in JSP:-

√ page directive

√ include directive

√ taglib directive

(I) what are Page directives?

Page directive is used to define page attributes the JSP file. Below is a sample of the same:-

<%@ page language="Java" import="java.rmi.*,java.util.*" session="true" buffer="12kb" autoFlush="true" errorPage="error.jsp" %>

To summarize some of the important page attributes:-

import :- Comma separated list of packages or classes, just like import statements in usual Java code.

session :- Specifies whether this page can use HTTP session. If set "true" session (which refers to the javax.servlet.http.HttpSession) is available and can be used to access the current/new session for the page. If "false", the page does not participate in a session and the implicit session object is unavailable.

buffer :- If a buffer size is specified (such as "50kb") then output is buffered with a buffer size not less than that value.

isThreadSafe :- Defines the level of thread safety implemented in the page. If set "true" the JSP engine may send multiple client requests to the page at the same time. If "false" then the JSP engine queues up client requests sent to the page for processing, and processes them one request at a time, in the order they were received. This is the same as implementing the javax.servlet.SingleThreadModel interface in a servlet.

errorPage: - Defines a URL to another JSP page, which is invoked if an unchecked runtime exception is thrown. The page implementation catches the instance of the Throwable object and passes it to the error page processing.

(I) what are include directives?

The include directive informs the JSP engine to include the content of the resource in the current JSP page. Below is the syntax for include statement.

```
<%@ include file="Filename" %>
```

Below is a code snippet which shows include directive in action

```
<html>
  <head>
    <title>Directive in action</title>
  </head>
    <%@ include file="/companyname.html" %>
  <body>
    <h1>Directive in action</h1>
  </body>
</html>
```

companyname.html contains the following:

```
<p>Koirala and Koirala Limited</p>
```

(I) Can you explain taglib directives?

Taglib are also termed as JSP tag extensions. They provide a way of encapsulating reusable functionality on JSP pages. One of the biggest drawbacks of scripting environments such as JSP is that it's easy to get carried away without thinking about how it will be maintained and grown in the future. For example, the ability to generate dynamic content by using Java code embedded in the page is a very powerful feature of the JSP specification. Custom tags allow such functionality to be encapsulated into reusable components. You can make write your reusable class in JAVA and call the same using XML tag.

There are four files which play an important role:-

√ Main class file which encapsulates the logic.

√ Tag library descriptor file.

√ Web.xml file which has the tag library descriptor file location.

√ Finally the JSP file which calls it.

Below is the image which shows the four files in one go.

Figure 4.16 : - taglib directive in action

The first file is the class file which will has the reusable code which will be called in the JSP file. The above class is also called as tag handler class. One of the important things to note is doStartTag() and doEndTag(). doStartTag is called when the JSP engine encounters an open tag and doEndTag is called when it encounters a closed tag.

The second important file is the tag descriptor file. This file maps the class name with a name which will be used to call this class.

The third file is the web.xml file. We need to define the tag library descriptor file location in web.xml file.

Finally is the JSP file which calls the class. There are two things to be noted in the JSP file first is the taglib which refers to the URI. Second is the custom tag which calls the datetime class.

> *Note: - You can find the above code sample in "JspTag" folder. Both the class as well as the JSP file is in the same folder. It displays date and time.*

(A) How does JSP engines instantiate tag handler classes instances?

JSP engines will always instantiate a new tag handler instance every time a tag is encountered in a JSP page. A pool of tag instances are maintained and reusing them where possible. When a tag is encountered, the JSP engine will try to find a Tag instance that is not being used and use the same and then release it.

(I) what's the difference between JavaBeans and taglib directives?

JavaBeans and taglib fundamentals were introduced for reusability. But following are the major differences between them:-

- √ Taglib are for generating presentation elements while JavaBeans are good for storing information and state.
- √ Use custom tags to implement actions and JavaBeans to present information.

(I) what are the different scopes an object can have in a JSP page?

There are four scope which an object can have in a JSP page:-

Page Scope

Objects with page scope are accessible only within the page. Data only is valid for the current response. Once the response is sent back to the browser then data is no more valid. Even if request is passed from one page to other the data is lost.

Request Scope

Objects with request scope are accessible from pages processing the same request in which they were created. Once the container has processed the request data is invalid. Even if the request is forwarded to another page, the data is still available though not if a redirect is required.

Session Scope

Objects with session scope are accessible in same session. Session is the time users spend using the application, which ends when they close their browser or when they go to another Web site. So, for example, when users log in, their username could be stored in the session and displayed on every page they access. This data lasts until they leave the Web site or log out.

Application Scope

Application scope objects are basically global object and accessible to all JSP pages which lie in the same application. This creates a global object that's available to all pages. Application scope variables are typically created and populated when an application starts and then used as read-only for the rest of the application.

(I) what are different implicit objects of JSP?

pageContext :- The PageContext object.

pageScope :- A Map of all the objects that have page scope.

requestScope :- A Map of all the objects that have request scope.

sessionScope :- A Map of all the objects that have session scope.

applicationScope :- A Map of all the objects that have application scope.

param :- A Map of all the form parameters that were passed to your JSP page (for example, the HTML <input name="ourName" type="text"/> is passed to your JSP page as a form parameter).

paramValues :- HTML allows for multiple values for a single form parameter. This is a Map of all the parameters, just like param, but in this object the values are an array containing all of the values for a given parameter in the event that there's more than one.

header :- A Map of all the request headers.

headerValues :- For the same reasons as paramValues, a headerValues object is provided.

cookie :- A Map of all the cookies passed to your JSP. The value returned is a Cookie object.

initParam :- A Map that maps context initialization parameter names to their parameter values.

(I) what are different Authentication Options available in servlets?

There are four ways of authentication:-

√ HTTP basic authentication

√ HTTP digest authentication

√ HTTPS client authentication

√ Form-based authentication

Let's try to understand how the above four ways work.

HTTP basic authentication

In HTTP basic authentication the server uses the username and password send by the client. The password is sent using simple base64 encoding but it's not encrypted.

HTTP digest authentication

HTTP digest authentication is same as HTTP basic authentication but the biggest difference is password is encrypted and transmitted using SHA or MD5.

HTTPS client authentication

HTTPS client authentication is based on HTTP over SSL. It requires that the end client should possess a PKC (Public Key Certificate). This verifies the browsers identity.

Form-based authentication

In FORM-based the web container invokes a login page. The invoked login page is used to collect username and password.

We will be seeing how to implement the above four using tomcat.

(A) Can you explain how do we practically implement security on a resource?

(A) How do we practically implement form based authentication?

Note: - We will answer this question from the perspective of tomcat server.

Below are the five important steps to implement security on a resource.

√ Define the URL pattern on which security will be applied.

√ Define the roles which are allowed to access the resource.

√ Define all the roles in the web application.

√ Define the authentication method.

√ Relate the users with roles.

We will try to understand the above steps in more detail using the figure below. In the below figure you can see the sequence of steps.

```
<web-app>

        <welcome-file-list>
            <welcome-file>index.html</welcome-file>
        </welcome-file-list>

        <security-constraint>

            <web-resource-collection>
                <web-resource-name>SecurePages</web-resource-name>
                <description>Security constraint for resources in the secur
                <url-pattern>/secure/*</url-pattern>
                <http-method>POST</http-method>
                <http-method>GET</http-method>
            </web-resource-collection>

            <auth-constraint>
                <description>only let the system user login </description>
                <role-name>admin</role-name>
            </auth-constraint>

            <user-data-constraint>
                <description>SSL not required</description>
                <transport-guarantee>NONE</transport-guarantee>
            </user-data-constraint>

        </security-constraint>
        <login-config>
            <auth-method>FORM</auth-method>
            <form-login-config>
                <form-login-page>/LoginForm.html</form-login-page>
                <form-error-page>/LoginError.html</form-error-page>
            </form-login-config>
        </login-config>
        <security-role>
            <description>The Secure ROLE</description>
            <role-name>admin</role-name>
        </security-role>

</web-app>


        <tomcat-users>
          <role rolename="tomcat"/>
          <role rolename="role1"/>
          <role rolename="manager"/>
          <role rolename="admin"/>
          <user username="tomcat" password="tomcat" roles="tomcat"/>
          <user username="role1" password="tomcat" roles="role1"/>
          <user username="both" password="tomcat" roles="tomcat,role1"/>
          <user username="interviewer" password="interviewer" roles="admin"/>
          <user username="admin" password="" roles="admin,manager"/>
        </tomcat-users>
```


Figure 4.17 : - Authentication steps in Web.xml

Define the URL pattern on which security will be applied

In this step we define the resource which we want to protect. This is defined in the web.xml file as shown in the figure above. In the <url-pattern> we have defined secure as the web application.

Define the roles which are allowed to access the resource

In step 2 we define which roles are allowed to access the resource. You can see in the above figure it's done by using <auth-constraint> tag in web.xml.

Define all the roles in the web application

In step 3 we define roles for the web application. We also need to mention the security role in <security-role>. Just a note the <auth-constraint> element specifies that application users must also be in the role of "Admin", and the <security-role> element defines that role.

Define the authentication method

This is the most important part defining which type of authentication we want to use. It's defined by using the <auth-method> tag. In this scenario we are using FORM authentication. You can also see sub tags <form-login-config> which define the main login page and the error page.

Relate the users with roles

Finally we need to define the actual users which can be attached to the roles. In the above figure you can see we have "interviewer" as user which is attached to admin role. Users can be defined in tomcat-users.xml file.

> *Note: - We have a sample application in formlogin folder which demonstrates how to execute the above steps practically. Copy the folder and paste it in webapps folder and execute the same to see how it works.*

(I) How do we authenticate using JDBC?

(I) Can you explain JDBCRealm?

A realm is a "database" of usernames, passwords, and user roles. When we say JDBCrealm we mean all these attributes are stored in table. In order to connect to JDBC you need to

make a context.xml and save the same in “META-INF “folder of the web application. Below is the context.xml snippet.

```
<Context path="/security" docBase="security" debug="0">
   <Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
        driverName="com.mysql.jdbc.Driver"
connectionURL="jdbc:mysql://localhost:3306/security?autoReconnect=true"
        connectionName="MyConn"
        connectionPassword="Password1"
        userTable="tblusers"
        userNameCol="username"
        userCredCol="password"
        userRoleTable="user_roles"
        roleNameCol="role_name" />
</Context>
```

(A) Can you explain how do you configure JNDIRealm?

Left to the users ? . That’s because it’s rarely asked but yes it’s asked.

(I) How did you implement caching in JSP?

OSCache is an open-source caching library that's available free of charge from the OpenSymphony organization (for more details visit http://www.opensymphony.com/oscache). OSCache has a set of JSP tags that make it easy to implement page caching in your JSP application.

Following are some Cache techniques it fulfills:-

Cache entry

An object that's stored into a page cache is known as a cache entry. In a JSP application, a cache entry is typically the output of a JSP page, a portion of a JSP page, or a servlet.

Cache key

A page cache is like a hash table. When you save a cache entry in a page cache, you must provide a cache key to identify the cache data. You can use keys like URI, other parameters like username, ipaddress to indentify cache data.

Cache duration

This is the period of time that a cache entry will remain in a page cache before it expires. When a cache entry expires, it's removed from the cache and will be regenerated again.

Cache scope

This defines at what scope the data is stored application or session scope.

```
<os:cache time="60">
    <%= new java.util.Date().toString() %></p>
</os:cache>
```

The above tag says that refresh after every 60 seconds the user requests data. So if user1 is requesting the page it will display fresh date and if an other user requests with in 60 seconds it will show same data. If any other user requests the page after 60 second he will again see refreshed date.

(B) What is the difference between Servletcontext and ServletConfig ?

ServletConfig contains configuration data for the servlet in the form of name and value pairs.Using the ServletConfigwe get reference to the ServletContext object. ServletContext gives the servlet access to information about its runtime environment such as web server logging facilities, version info, URL details, web server attributes etc.

(I) How do we prevent browser from caching output of my JSP pages?

You can prevent pages from caching JSP pages output using the below code snippet.

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

(I) Can we explicitly destroy a servlet object?

No we can not destroy a servlet explicitly its all done by the container. Even if you try calling the destroy method container does not respond to it.

Distribution Partner

Do you have a news group or website where you want to distribute this PDF free. Contact at shiv_koirala@yahoo.com we will put your logo and send you a complete zipped file which you can host at your site to increase user visits. Be our partner and increase your user visits in your website. We do not take any charge from our partner to distrubute these sample copies. But yes the contents of this PDF can not be modified. If you are our partner you get regular updates of our interview question releases.

Illegal distributions of this PDF will be taken seriously. Just send a mail and be our distribution partner with your website logo proudly do not distribute illegally.

www.questpond.com