5

UNIT

Servlets and Java Server Pages (JSP)

CONTENTS

PART-1

Servlets : Servlet Overview and Architecture, Interface Servlet and the Servlet Life Cycle.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. **What is servlet ? Explain its life cycle. Give its characteristics.**

Answer

Servlet :

1. Servlets are simple Java programs that run on the servers.
2. Servlets are most commonly used with HTTP. So, servlets are also called as HTTP servlet.
3. Servlet can process and store the data submitted by an HTML form.
4. Servlets are useful for providing the dynamic contents.

Life cycle of Servlet :

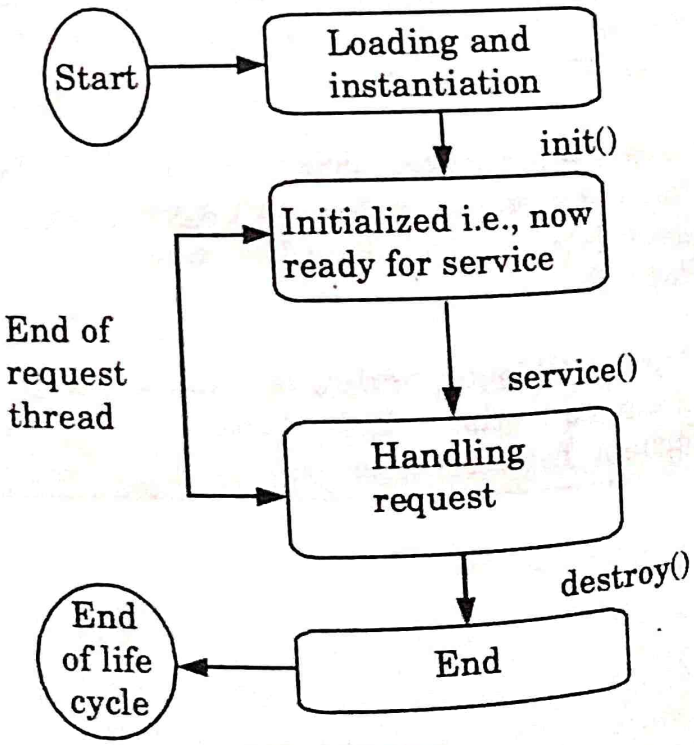


Fig. 5.1.1.

Stages of the Servlet life cycle :

1. **Loading a Servlet :**
   a. The first stage of the Servlet lifecycle involves loading and initializing the Servlet by the Servlet container.
   b. The Servlet container performs two operations in this stage :
      i. **Loading :** Loads the Servlet class.
      ii. **Instantiation :** Creates an instance of the Servlet. To create a new instance of the Servlet, the container uses the no-argument constructor.
2. **Initializing a Servlet :**
   a. After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object.
   b. The container initializes the Servlet object by invoking the Servlet.init(ServletConfig) method which accepts ServletConfig object reference as parameter.
3. **Handling request :**
   a. After initialization, the Servlet instance is ready to serve the client requests.
   b. The Servlet container performs the following operations when the Servlet instance is located to service a request :
      i. It creates the ServletRequest and ServletResponse objects.
      ii. After creating the request and response objects it invokes the Servlet.service(ServletRequest, ServletResponse) method by passing the request and response objects.
4. **Destroying a Servlet :**
   a. When a Servlet container decides to destroy the Servlet, it performs the following operations,
      i. It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
      ii. After currently running threads have completed their jobs, the Servlet container calls the destroy() method on the Servlet instance.
   b. After the destroy() method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.

Characteristics of servlet :

1. Servlet operates on input data that is encapsulated in a request object.
2. Servlet responds to a query with data encapsulated in a response object.
3. Servlet can call EJB components to perform business logic functions.
4. Servlet can call JSPs to perform page layout functions.
5. Servlet can call other servlets.

Que 5.2. **Explain the life cycle of servlet. Also write a servlet for displaying a string "HELLO WORLD!"** **AKTU 2017-18, Marks 10**

Answer

Life cycle of servlet : Refer Q. 5.1, Page 5–2D, Unit-5.

Servlet for displaying "HELLO WORLD" :

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
// Extend HttpServlet class
public class HelloWorld extends HttpServlet {
private String message;
public void init() throws ServletException {
// Do required initialization
message = "HELLO WORLD";
}
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
// Set response content type
response.setContentType("text/html");
// Actual logic goes here.
PrintWriter out = response.getWriter();
out.println("<h1>" + message + "</h1>");
}
public void destroy() {
// do nothing.
}
}
```

Que 5.3. **What do you mean by Common Gateway Interface (CGI)? How does CGI work ?**

Answer

1. CGI is an acronym for the Common Gateway Interface which is a standard protocol for running programs within a web server.

2. The CGI protocol allows external program to interface with programs such as database management software and to access the networking facilities provided by the HTTP server software.
3. CGI programs are dynamic and the state of their variables alters as they execute.
4. CGI allows a WWW server to provide information to WWW clients that would otherwise not be available to those clients.

Working of CGI :

1. A reader sends a URL that causes the AOLserver to use CGI to run a program.
2. The AOLserver passes input from the reader to the program and output from the program back to the reader. CGI acts as a "gateway" between the AOLserver and the program.
3. The program run by CGI can be any type of executable file on the server platform. For example, we can use C, C++, Perl, Unix shell scripts.

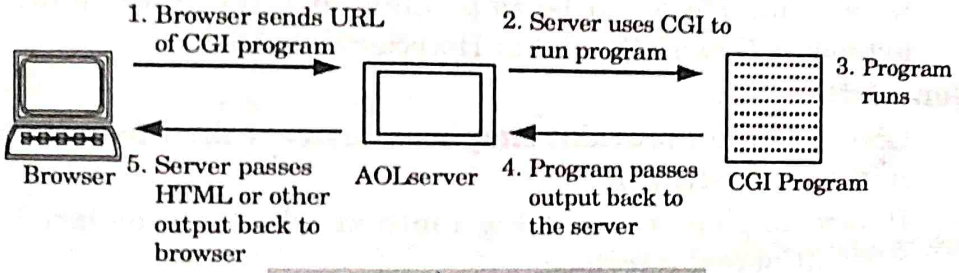


Fig. 5.3.1. Working of CGI.

Que 5.4. What is servlet ? Explain its life cycle. Illustrate some characteristics of servlet. How does servlet score over CGI ?

AKTU 2015-16, Marks 10

Answer

Servlet, its lifecycle and characteristics of servlet : Refer Q. 5.1, Page 5–2D, Unit-5.

Java servlets have following advantage over CGI and other API's :

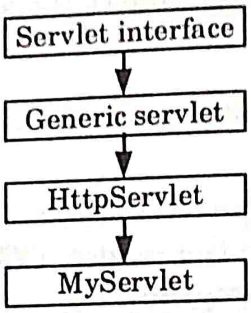
i. **Platform independence :** Java servlets are pure Java program, so it is platform independent. It can run on any servlet enabled web server.
ii. **Performance :** In case of servlets, initialization takes place very first time it receives a request and remains in memory till times out or server shut downs. This helps us to develop high speed data driven websites.
iii. **Extensibility :** Java servlets are developed in Java which is robust, well-designed and object-oriented language which can be extended into new objects.
iv. **Safety :** Java servlet provides a very good safety features like memory management, exception handling features and emerged as a very powerful web server extension.

v. **Secure :** Servlets are server-side components, so it inherits the security provided by the web server.

Que 5.5. Explain servlet architecture.

Answer

Servlet architecture includes :



1. **Servlet interface :**
   a. To write a servlet we need to implement Servlet interface.
   b. Servlet interface can be implemented directly or indirectly by extending GenericServlet or HttpServlet class.
2. **GenericServlet :**
   a. GenericServlet provides simple versions of the lifecycle methods init( ) and destroy( ).
   b. It also implements the log method which is declared in the ServletContext interface.
3. **HttpServlet :**
   a. HttpServlet is an abstract class present in javax.servlet.http package and has no abstract methods.
   b. It extends GenericServlet class.
   c. When the servlet container uses HTTP protocol to send request, then it creates HttpServletRequest and HttpServletResponse objects.
4. **MyServlet :** MyServlet is the combination of servlet interface, genericServlet and HttpServlet.

Que 5.6. Write short notes on servlet interface.

Answer

1. Servlet interface provides common behaviour to all the servlets.
2. Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
3. Servlet interface defines methods that are implemented by all servlets.
4. Following are five methods defined in servlet interface :

S.No.	Method	Description
1.	public void init(ServletConfig config)	It initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
2.	public void service(ServletRequest request,ServletResponse response)	It provides response for the incoming request. It is invoked at each request by the web container.
3.	public void destroy()	It is invoked only once and indicates that servlet is being destroyed.
4.	public ServletConfig getServletConfig()	It returns the object of ServletConfig.
5.	public String getServletInfo()	It returns information about servlet such as writer, copyright, version etc

Que 5.7. Explain how servlet works.

Answer

1. When the web server starts, the servlet container deploys and loads all the servlets.
2. Servlet container creates ServletContext object which act as an interface and defines the set of methods that a servlet can use to communicate with the servlet container.
2. Once the servlet is loaded, the servlet container creates the instance of servlet class. For each instantiated servlet, its init() method is invoked.
3. Client (user browser) sends an Http request to web server on a certain port.
4. Then, the servlet container creates HttpServletRequest and HttpServletResponse objects.
5. The HttpServletRequest object provides the access to the request information and the HttpServletResponse object allows us to change the http response before sending it to the client.
6. The servlet container produce a new thread that calls service() method for each client request.
7. The service() method dispatches the request to the correct handler method based on the type of request.

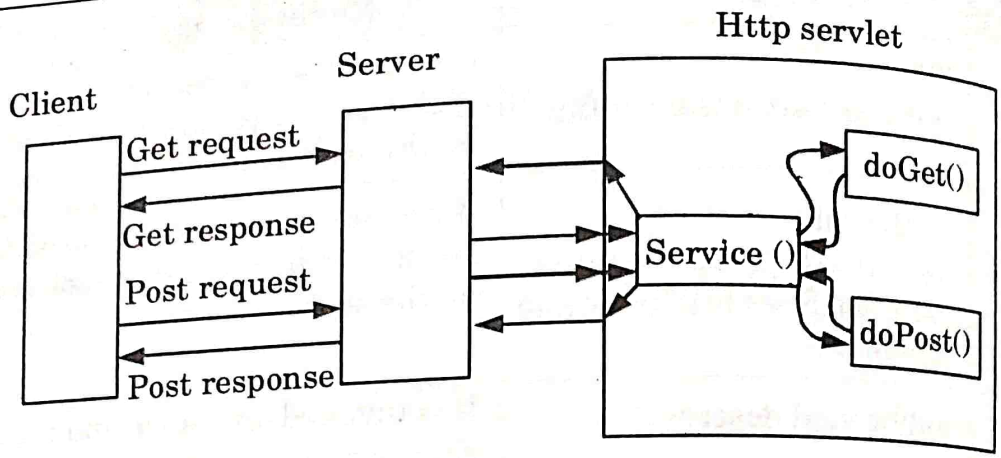


Fig. 5.7.1.

8. When servlet container shuts down, it unloads all the servlets and calls destroy() method for each initialized servlets.

Que 5.8. Explain servlets with its life cycle. How its life cycle is different from the life cycle of JSP ? Explain with an example.

AKTU 2019-20, Marks 07

Answer

Servlets with its life cycle : Refer Q. 5.1, Page 5–2D, Unit-5.

Difference :

1. In servlet life cycle, the servlet object is created first.
2. The init() method is invoked by the servlet container and the servlet is initialized by its arguments.
3. Servlet's service() method is invoked next. At the end, the destroy() method is invoked.
4. In case of a Java Server Page life cycle, the .jsp is converted into .class file which is a servlet and then follows the process of the servlet. In other words, the .jsp is translated into servlet and the functionality is same as that of the servlet.

JSP lifecycle example :

Demo.jsp :

```
<html>
<head>
<title>Demo JSP</title>
</head>
<%
int demvar=0;%>
<body>
Count is :
<% Out.println(demovar++); %>
<body>
</html>
```

Demo JSP Page is converted into demo_jsp.servlet in the below code.

```
Public class demp_jsp extends HttpServlet{
Public void _jspservice(HttpServletRequest request,
HttpServletResponse response) Throws IOException, ServletException
{
PrintWriter out = response.getWriter();
response.setContentType("text/html");
out.write ("<html><body>");
int demovar=0;
out.write("Count is:");
out.print(demovar++);
out.write("</body></html>");
}
}
```

5. The web container calls the init() method only once after creating the servlet instance.
6. The init() method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface.
7. Syntax of the init() method :
   public void init(ServletConfig config) throws ServletException
8. A new thread is then gets created, which invokes the_jspService() method, with a request (HttpServletRequest) and response (HttpServletResponse) objects as parameters shown below :

```
[code language="java"]
void _jspService( HttpServletRequest req, HttpServletResponse res)
{
//code goes here
}[/code]
```

9. Invokes the jspDestroy() method to destroy the instance of the servlet class. code :

```
[code language="java"]
public void jspDestory()
{
//code to remove the instances of servlet class
}
```

PART-2

Handling HTTP Get Requests, Handling Post Requests, Redirecting Requests to Other Resources.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.9. Explain handling in HTTP get and post request in servlet with example.

Answer

1. To handle HTTP requests in a servlet, extend the HttpServlet class and override the servlet methods that handle the HTTP requests that our servlet supports.
2. The methods that handle these requests are doGet() and doPost().

Handling Get requests :

1. Handling Get requests involves overriding the doGet() method.

For example :

```
public class BookDetailServlet extends HttpServlet {
public void doGet (HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
...
// set content-type header before accessing the Writer
response.setContentType("text/html");
PrintWriter out = response.getWriter();
// then write the response
out.println("<html>" +
"<head><title>Book Description</title></head>" +
...);
//Get the identifier of the book to display
String bookId = request.getParameter("bookId");
if (bookId != null) {
// and the information about the book and print it
...
}
out.println("</body></html>");
out.close();
}
...
}
```

2. The servlet extends the HttpServlet class and overrides the doGet() method.
3. Within the doGet() method, the getParameter () method gets the servlet's expected argument.

4. To respond to the client, the example doGet() method uses a Writer from the HttpServletResponse object to return text data to the client.
5. At the end of the doGet() method, after the response has been sent, the Writer is closed.

Handling Post requests :

1. Handling Post requests involves overriding the doPost() method.

For example :

```
public class ReceiptServlet extends HttpServlet {
public void doPost(HttpServletRequest request,
 HttpServletResponse response)
throws ServletException, IOException
{
...
// set content type header before accessing the Writer
response.setContentType("text/html");
PrintWriter out = response.getWriter();
// then write the response
out.println("<html>" +
"<head><title> Receipt </title>" +
...);
out.println("<h3>Thank you for purchasing your books from us" +
      request.getParameter("cardname") +
           ...);
out.close();
}
...
}
```

2. The servlet extends the HttpServlet class and overrides the doPost() method.
3. Within the doPost() method, the getParameter() method gets the servlet's expected argument.
4. To respond to the client, the example doPost() method uses a Writer from the HttpServletResponse object to return text data to the client.
5. At the end of the doPost() method, after the response has been set, the Writer is closed.

Que 5.10. Explain RequestDispatcher. Also describe different ways to get the object of RequestDispatcher.

Answer

1. The RequestDispatcher is a interface which provides the facility of dispatching the request to another resource such as HTML, servlet or JSP.
2. This interface can also be used to include the content of another resource also.
3. It is one of the ways of servlet collaboration.

Methods of RequestDispatcher interface :

1. **Public void forward(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException :** It forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. **Public void include(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException :** It includes the content of a resource (servlet, JSP page, or HTML file) in the response.

There are three ways to get RequestDispatcher object in the servlet :

1. **By calling the getRequestDispatcher(String path) method on ServletRequest :** It returns a RequestDispatcher object that wraps the resource located at the given path. This method will return null if the ServletContext cannot return a RequestDispatcher.
2. **By getRequestDispatcher(String path) method on ServletContext :** It returns a RequestDispatcher object that wraps the named servlet or JSP page. Names can be defined for servlets and JSP pages in the web application. If the ServletContext cannot return a RequestDispatcher, this method will return null.
3. **By calling getNamedDispatcher(servlet logical name) on ServletContext object :** It returns a RequestDispatcher object that acts as a wrapper for the resource located at the specified context-relative path.

PART-3

Session Tracking, Cookies, Session Tracking with HTTP Session.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.11. Write short notes on session tracking.

Answer

1. Session tracking is a way to maintain state (data) of a user. It is also known as session management in servlet.
2. It is used to recognize the particular user.
3. Each time user requests to the server, server treats the request as the new request and maintain the state of a user using session tracking techniques to recognize a particular user.
4. HTTP is stateless that means each request is considered as the new request. It is shown in the Fig. 5.11.1.

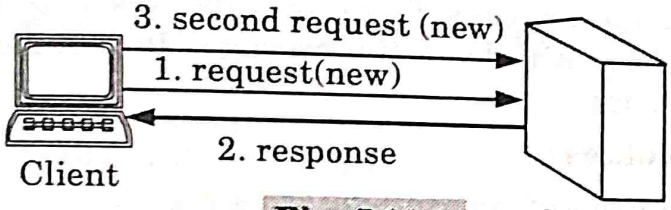


Fig. 5.11.1.

5. There are four techniques used in session tracking :
   a. Cookies
   b. Hidden Form Field
   c. URL Rewriting
   d. HttpSession

Que 5.12. Explain cookies with its advantages and disadvantages. Also give its type.

Answer

1. Cookie is a key value pair of information sent by the server to the browser which is saved in the client computer.
2. A cookie is a small piece of information that is persisted between the multiple client requests.
3. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

Working of cookie :

1. Client sends a request to the server.
2. Cookie is added with request from the servlet and stored in the cache of the browser.
3. After that if response is sent by the server, cookie is added with it by default.
4. Server can identify the client using the cookie.

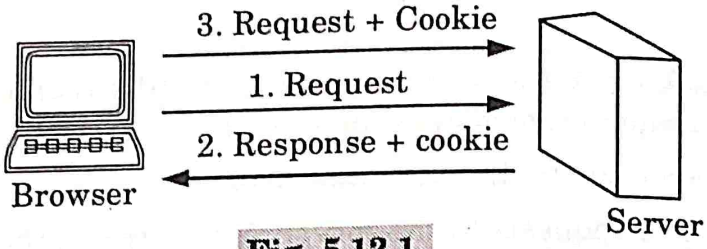


Fig. 5.12.1.

There are two types of cookies in servlet :

1. **Non-persistent cookie :** It is valid for single session only. It is removed each time when user closes the browser.
2. **Persistent cookie :** It is valid for multiple sessions. It is not removed each time when user closes the browser. It is removed only if user logout or sign out.

Advantage of cookies :

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client-side.

Disadvantage of cookies :

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in cookie object.

Que 5.13. Write short notes on session tracking with HttpSession.

Answer

1. The HttpSession object is used for session tracking.
2. A session contains information specific to a particular user across the whole application.
3. When a user enters into a website (or an online application) for the first time HttpSession is obtained via request.getSession(), the user is given a unique ID to identify his session. This unique ID can be stored into a cookie or in a request parameter.

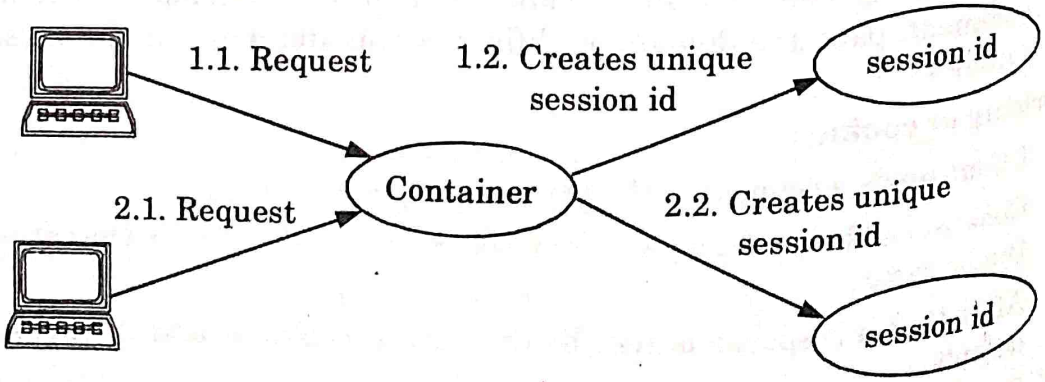


Fig. 5.13.1.

The HttpServletRequest interface provides two methods to get the object of HttpSession :

1. **public HttpSession getSession() :** It returns the current session associated with this request, or if the request does not have a session, creates one.
2. **public HttpSession getSession(boolean create) :** It returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

Commonly used methods of HttpSession interface are :

1. **public String getId() :** It returns a string containing the unique identifier value.
2. **public long getCreationTime() :** It returns the time when this session was created, measured in milliseconds.
3. **public long getLastAccessedTime() :** It returns the last time the client sent a request associated with this session.
4. **public void invalidate() :** It invalidates this session then unbinds any objects bound to it.

Que 5.14. What is difference between session and cookies ? Write a servlet program for servlet login and logout using cookies.

AKTU 2018-19, Marks 07

Answer

S. No.	Session	Cookies
1.	Sessions are stored in server.	Cookies are stored in the user's browser.
2.	A session is available as long as the browser is opened. User can not disable the session.	Cookies can keep information in the user's browser until deleted by user or set as per the timer.
3.	It will be destroyed if we close the browser.	It will not be destroyed even if we close the browser.
4.	It can store any object.	Cookies can only store string.
5.	Session cannot be same for future reference.	We can save cookies for future reference.

Program :

Index.html :

```
<!DOCTYPE html><html><head>
```

```
<meta charset="ISO-8859-1">
<title>Servlet Login Example</title>
</head><body>
<h1>Welcome to Login App by Cookie</h1>
<a href="login.html">Login</a>|
<a href="LogoutServlet">Logout</a>|
<a href="ProfileServlet">Profile</a></body></html>
```

link.html :

```
<a href="login.html">Login</a> |
<a href="LogoutServlet">Logout</a> |
<a href="ProfileServlet">Profile</a>
<hr>
```

login.html :

```
<form action="LoginServlet" method="post">
Name:<input type="text" name="name"><br>
Password:<input type="password" name="password"><br>
<input type="submit" value="login"></form>
```

LoginServlet.java :

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LoginServlet extends HttpServlet {
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
request.getRequestDispatcher("link.html").include(request, response);
String name=request.getParameter("name");
String password=request.getParameter("password");
if(password.equals("admin123")){
```

```
out.print("You are successfully logged in!");
out.print("<br>Welcome, "+name);
Cookie ck=new Cookie("name",name);
response.addCookie(ck);
}else{
out.print("sorry, username or password error!");
request.getRequestDispatcher("login.html").include(request, response);}
out.close();}}
```

LogoutServlet.java :

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LogoutServlet extends HttpServlet {
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
request.getRequestDispatcher("link.html").include(request, response);
Cookie ck=new Cookie("name","");
ck.setMaxAge(0);
response.addCookie(ck);
out.print("you are successfully logged out!");}}
```

Que 5.15. **Create a form in HTML taking account number from user as input then write a servlet program receiving this from data and connect it with database by using JDBC. Then send the current account balance of user stored in specific database back to user as response. Also, mention all the assumed required data like table name, database name and fields name etc.**

AKTU 2017-18, Marks 10

Answer

Let us assume a table "account_detail" which contains the details about the Account number.

```
create table account_detail
(
ac_no number,
name VARCHAR2(40),
address VARCHAR2(40),
balance number,
CONSTRAINT "account_detail_pk" PRIMARY KEY ("ac_no") ENABLE
)
```

Index.html :

This page get the account number from the user and forwards this data to servlet which is responsible to show the records based on the given account number.

```
<html>
<body>
<form action="servlet/Search">
Enter A/c No.:<input type="text" name="ac"/></br>
<input type="submit" value="search"/>
</form>
</body>
</html>
```

Search.java :

This is the servlet file which gets the input from the user and maps this data with the database and prints the record for the matched data. In this page, we are displaying the column name of the database along with data. So, we are using ResultSetMetaData interface.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class Search extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
```

```
PrintWriter out = response.getWriter();
String rollno=request.getParameter("ac");
int ac=Integer.valueOf(ac_no);
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle"); PreparedStatement
ps=con.prepareStatement("select * from result where ac_no =?");
ps.setInt(1,ac);
out.print("<table width=50% border=1>");
out.print("<caption>Result:</caption>");
ResultSet rs=ps.executeQuery();
/* Printing column names */
ResultSetMetaData rsmd=rs.getMetaData();
int total=rsmd.getColumnCount();
out.print("<tr>");
for(int i=1;i<=total;i++)
{
out.print("<th>"+rsmd.getColumnName(i)+"</th>");
}
ut.print("</tr>");
/* Printing result */
while(rs.next())
{
out.print("<tr><td>"+rs.getInt(1)+"</td><td>"+rs.getString(2)+
"</td><td>"+rs.getString(3)+"</td><td>"+rs.getString(4)+
"</td></tr>");
}
out.print("</table>");
}catch (Exception e2) {e2.printStackTrace();}
finally{out.close();}
}
}
```

web.xml file :

This is the configuration file which provides information of the servlet to the container.

```
<web-app>
<servlet>
<servlet-name>Search</servlet-name>
<servlet-class>Search</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Search</servlet-name>
```

```
<url-pattern>/servlet/Search</url-pattern>
</servlet-mapping>
</web-app>
```

PART-4

Java Server Page (JSP) : Introduction, Java Server Page Overview, A First Java Server Page Example.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.16. What do you mean by JSP ? Explain the architecture of JSP. How JSP provides better performance ?

Answer

1. Java Server Pages (JSP) is a technology that helps software developers to create dynamically generated web pages based on HTML, XML or other document types.
2. JSP supports both scripting and element based dynamic content, and allows developers to create their own tag libraries.
3. JSP pages are compiled for efficient server processing.
4. JSP is platform independent. So, it can be easily upgraded or switched without affecting JSP based applications.

JSP architecture :

1. Java Server Pages are part of a 3-tier architecture.
2. A server (application or web server) supports the Java Server Pages.
3. This server will act as a mediator between the client browser and a database.
4. The following diagram shows the JSP architecture :

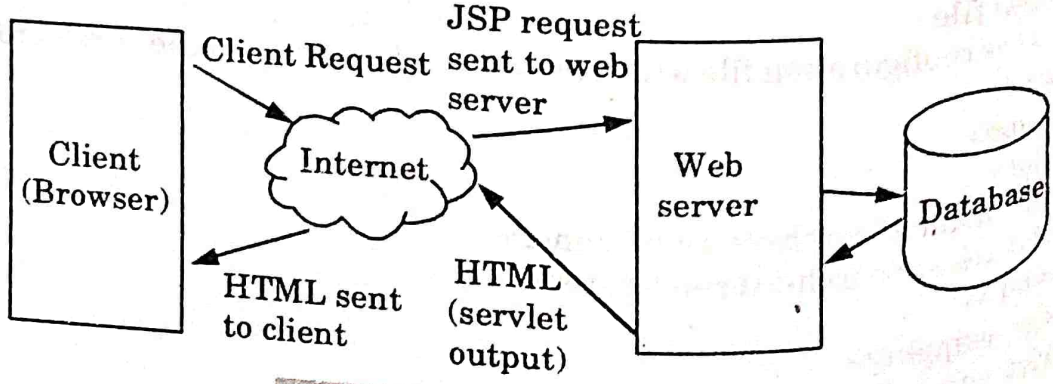


Fig. 5.16.1. JSP architecture.

a. The user goes to a JSP page and makes the request via internet in web browser.

b. The JSP request is sent to the web server.

c. Web server accepts the requested .jsp file and passes the JSP file to the JSP Servlet Engine.

d. If the JSP file has been called the first time then the JSP file is parsed otherwise servlet is instantiated. The next step is to generate a servlet from the JSP file. The generated servlet output is sent via the Internet from web server to user's web browser.

d. Now in last step, HTML results are displayed on the user's web browser.

JSP provides better performance : JSP allows to embed the Java code directly in the HTML file and generates the contents dynamically. This helps to enhance the performance of JSP.

Que 5.17. What is JSP ? What are the advantages of JSP over various server-side programs ?

Answer

JSP : Refer Q. 5.16, Page 5–20D, Unit-5.

Advantages of JSP over other server-side programs are :

1. JSP is platform independent whereas other server-side programs are not.
2. JSP code is portable but other server-side programs are not.
3. JSP can be used for large application but other server-side programs are used for small application.
4. JSP uses static type checking while other uses dynamic type checking.

Que 5.18. Explain JSP processing.

Answer

1. In JSP processing, the JSP engine compiles the servlets upto an executable class and forwards the original request to the servlet engine and execute it on the web server.
2. JSP pages can be processed using JSP container only.
3. Following are the steps that need to be followed while processing the request for JSP page :
   a. Client makes a request for required JSP page to the server.
   b. The server must have JSP container so that JSP request can be processed.

c. On receiving request the JSP container searches and then reads the desired JSP page.

d. Then JSP page is converted to corresponding servlet. Basically any JSP page is a combination of template text and JSP element.

e. Every template text is translated into corresponding println statement.

Que 5.19. **Write note on Tomcat server.** **AKTU 2015-16, Marks 05**

OR

Explain in detail the Tomcat server. **AKTU 2016-17, Marks 10**

Answer

1. Tomcat is a web server and servlet container that is used to deploy and serve Java web application.
2. Tomcat server is a Java-capable HTTP server, which could execute special Java programs known as "Java servlet" and "Java Server Pages".
3. Tomcat can operate as a standalone web server.
4. It can operate as an out-of-process servlet container for some web servers, such as Apache.
5. For other web servers, such as IIS (Internet Information Services), it can operate as an in-process servlet container.
6. Tomcat server runs on a specific TCP post from a specific IP address.
7. The default TCP port number for HTTP protocol is 80, which is used for the production HTTP server.
8. To test HTTP server, we can choose any unused port number between 1024 and 65535.
9. There are two important environment variables to set before running Tomcat :
   a. Set CATALINA_HOME to the root of Tomcat directory.
   b. Set JAVA_HOME to the root directory of Java JDK or JRE.

Que 5.20. **What are the steps for running JSP program in Tomcat server ?**

Answer

Steps for running JSP program in Tomcat server are :

1. Create a JSP code and save it using the filename extension.jsp.
2. Copy this JSP file to the directory named webapps. This directory is present within the Tomcat directory.
3. Start the Tomcat server by typing the command startup.

4. Open some suitable web browser type the path for JSP code with the prefix http://localhost. Localhost is the default DNS for Tomcat web server.

Que 5.21. **Write a simple JSP page for displaying the message : "This is my first JSP page !!!".**

Answer

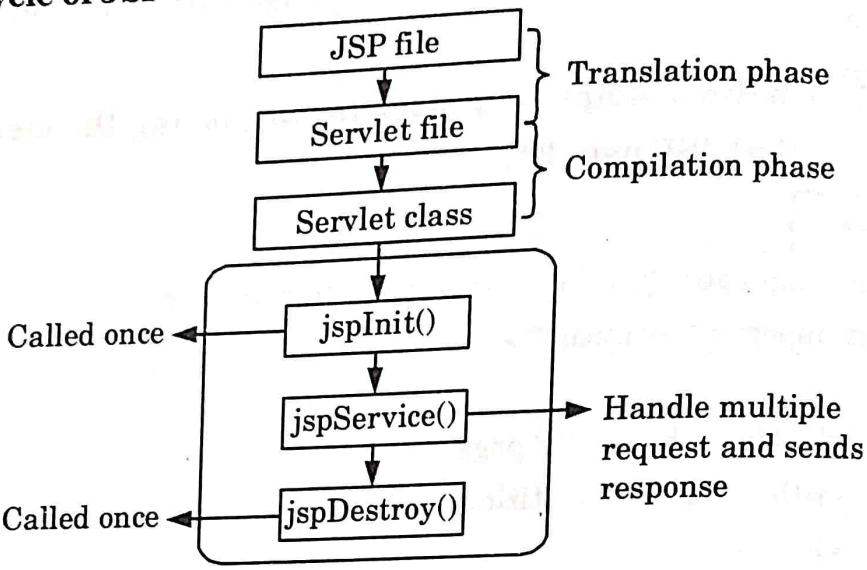
```
<%@ page language = "java" contentType = "test/html" %>
<%@ page import = "java.util.*">
<html>
        <!-- This is basic JSP page -->
        <title> JSP Demo </title>
        <body>
        <%--Displaying the message on the browser --%>
        <% out.println("This my first JSP page!!!"); %>
    </body>
</html>
```

Que 5.22. **Compare JSP and Servlet. Explain the life cycle of a JSP page with a suitable diagram. Also list any five action tags used in JSP.** **AKTU 2018-19, Marks 07**

Answer

S. No.	JSP	Servlet
1.	JSP is protocol dependent.	Servlet is protocol independent.
2.	It can handle only HTTP and HTTPS protocol.	It can handle any type of protocol *i.e.*, FTP, HTTP.
3.	Time taken to generate response for first request is more.	Time taken to generate response for first request is less.
4.	JSP is a scripting language which can generate dynamic response.	Servlet is a Java program which can generate dynamic response.
5.	It is easy to code.	It is not easy to code.

Life cycle of JSP :



1. **Translation of JSP page to Servlet :** This is the first step of JSP life cycle. This translation phase deals with syntactic correctness of JSP. Here test.jsp file is translated to test.java.
2. **Compilation of JSP page :** Here the generated java servlet file (test.java) is compiled to a class file (test.class).
3. **Class loading :** Servlet class which has been loaded from JSP source is now loaded into container.
4. **Instantiation :** Here instance of the class is generated. The container manages one or more instance by providing response to requests.
5. **Initialization :** jspInit() method is called only once during the life cycle immediately after the generation of servlet instance from JSP.
6. **Request processing :** jspService() method is used to serve the raised requests by JSP. It takes request and response object as parameters. This method cannot be overridden.
7. **JSP cleanup :** In order to remove the JSP from use by the container or to destroy method for servlets jspDestroy() method is used. This method is called once, if we need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

Five action tags used in JSP are :

1. <JSP : use Bean>
2. <JSP : set Property>
3. <JSP : include>
4. <JSP : attribute>
5. <JSP : body>

Que 5.23. How we deploy the Servlets on Tomcat Web Container ? Also explain how we change the default number of Tomcat Container ?

Answer

To create a Servlet applications do the following steps :

1. Create directory structure for our application.
2. Create a Servlet.
3. Compile the Servlet.
4. Create deployement descriptor for our application.
5. Start the server and deploy the application.

Following steps are required to change the default port of Tomcat container :

1. Stop Tomcat server if it is already running.
2. Open <Tomcat_Home>/conf/server.xml file for editing.
3. Look for port *8080* in the xml file and replace with any available port.
4. If there is other Tomcat running on the same server, we have to change SHUTDOWN port as well.
5. Start Tomcat server.

Que 5.24. JSP is an extension of Servlets not replacement, Justify ? What problems of Servlets technology can JSP is suppose to solve ?

Answer

JSP is an extension of servlet not replacement. This statement can be justified by following advantages of JSP :

1. JSP is not a replacement of Servlets but extension of Servlets, as coding decreases more than half.
2. In JSP, static code and dynamic code are separated.
3. JSP needs no compilation by the programmer.
4. In JSP, presentation layer and business logic layer can be separated with the usage of Java Beans.

Problem of Servlet technology solved by JSP are :

1. Difficult to code.
2. Do not allow parsing and decoding HTML headers.
3. It cannot be integrated with other backend services.
4. It does not manage cookies.
5. Do not allow reading and sending HTML headers.

PART-5

Implicit Objects, Scripting, Standard Actions, Directives, Custom Tag Libraries.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.25. Explain implicit objects available in JSP with example.

AKTU 2017-18, Marks 10

Answer

Different types of implicit objects in JSP are :

1. **Application object :** The application object has an application scope and contains a reference to the instance of a class that implements the javax.servlet.ServletContext interface that represents the application.
2. **Config object :**
   a. The config object has a page scope. This object implements the javax.servlet.ServletConfig interface.
   b. The config object gives access to configuration data for initializing the JSP.
3. **Session object :**
   a. HttpSession class represents the current session of the JSP page.
   b. It represents the scope of this session, and it is useful in order to keep attributes and values and providing them in different JSP pages of same application.
4. **Out object :**
   a. The out object also has a page scope. Out object is an instance of javax.servlet.jsp.JspWriter class.
   b. By using this object the text is added to the response message body.
5. **Page object :**
   a. Page object is an instance of java.lang.Object class.
   b. The page object is a reference to the current instance of the JSP.

Example :

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
pageEncoding="US-ASCII"%>
<%@ page import="java.util.Date" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-
ASCII">
```

```
<title>Index JSP Page</title>
</head>
<body>
<%-- out object example --%>
<h4>Hi There</h4>
<strong>Current Time is</strong>: <% out.print(new Date());
%><br><br>
<%-- config object example --%>
<strong>User init param value
</strong>:<%=config.getInitParameter("User") %><br><br>
<%-- application object example --%>
<strong>User context param value</strong>:
<%=application.getInitParameter("User") %><br><br>
<%-- session object example --%>
<strong>User Session ID</strong>:<%=session.getId() %><br><br>
<%-- page object example --%>
<strong>Generated Servlet Name</strong>:
<%=page.getClass().getName() %>
</body>
</html>
```

Que 5.26. **What are Java Beans ? Why they are used ? Write a JSP page and use an existing Java Beans in JSP page by using the standard action. Write the program with describing the output ?**

AKTU 2017-18, Marks 10

OR

What are standard actions in JSP ? Illustrate with example.

AKTU 2017-18, Marks 10

Answer

Java Beans and why they are used : Refer Q. 4.4, Page 4–5D, Unit-4.

Different types of standard action tags used in JSP are :

1. **<jsp:useBean> :**
   a. This action associates an instance of a Java Bean defined with a given scope and ID, through a newly declared scripting variable of the same ID.
   b. The <jsp:useBean> action is very flexible.
   c. Its exact semantics depends on the values of the given attributes.
   d. The attributes for the <jsp:useBean> are : id, scope, class, beanName, type.
2. **<jsp:setProperty> :**
   a. This action helps to integrate Java Beans into JSPs.
   b. It sets the value of a Beans property.
   c. This action has the attributes name, property, param, value.

3. **<jsp:getProperty> :**
   a. This action gets a property value from a Java Beans component and adds it to the response.
   b. Attribute of this action are : name, property.
4. **<jsp:include> :**
   a. This action provides a mechanism for including additional static and dynamic resources in the current JSP page.
   b. The attributes for this action are : page, flush.
5. **<jsp:attribute> :** This action is used to set the value of an action attribute based on the body of this element.
6. **<jsp:body> :**
   a. This action is used to set the action element body based on the body of this statement.
   b. It is required when the action element body contains <jsp:attribute> action element.
7. **<jsp:element> :** It dynamically generates an XML element, optionally with attributes and a body defined by nested <jsp:attribute> and <jsp:body> actions.
8. **<jsp:text> :** This action is used to encapsulate template text that should be used in JSP pages written as XML documents.

Example of <jsp:include> Action :

```
<html>
<head>
<title>The include Action Example</title>
</head>
<body>
<center>
<h2>The include action Example</h2>
<jsp:include page = "date.jsp" flush = "true" />
</center>
</body>
</html>
```

Output :

The include action Example
Today's date: 12-june-2018 14:54:22

Example of <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> actions :

Let us define a test bean that will further be used :

```
/* File: TestBean.java */
package action;
public class TestBean {
private String message = "No message specified";
public String getMessage() {
return(message);
}
public void setMessage(String message) {
```

```
this.message = message;
}
}
```

Now use the following code in main.jsp file. This loads the bean and sets/gets a simple String parameter.

```
<html>
<head>
<title>Using JavaBeans in JSP</title>
</head>
<body>
<center>
<h2>Using JavaBeans in JSP</h2>
<jsp:useBean id = "test" class = "action.TestBean" />
<jsp:setProperty name = "test"  property = "message"
value = "Hello JSP..."/>
<p>Got message....</p>
<jsp:getProperty name = "test" property = "message"/>
</center>
</body>
</html>
```

Output :

Using JavaBeans in JSP
Got message....
Hello JSP...

Example of <jsp:element>, <jsp:attribute> and <jsp:body> actions :

```
<%@page language = "java" contentType = "text/html"%>
<html xmlns = "http://www.w3.org/1999/xhtml"
xmlns:jsp = "http://java.sun.com/JSP/Page">
<head><title>Generate XML Element</title></head>
<body>
<jsp:element name = "xmlElement">
<jsp:attribute name = "xmlElementAttr">
Value for the attribute
</jsp:attribute>
<jsp:body>
Body for XML element
</jsp:body>
</jsp:element>
</body>
</html>
```

Example of <jsp:text> action :

```
<jsp:text><![CDATA[<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">]]></jsp:text>
<head><title>jsp:text action</title></head>
<body>
```

```
<books><book><jsp:text>
Welcome to JSP Programming
</jsp:text></book></books>
</body>
</html>
```

Que 5.27. **What do you mean by JSP processing ? How JSP pages are handled ? Explain various JSP directives with suitable examples.**

AKTU 2015-16, Marks 10

OR

What are JSP directives ? Explain various types of directives with example.

AKTU 2017-18, Marks 10

Answer

JSP processing : Refer Q. 5.18, Page 5–21D, Unit-5.

JSP page handling :

Error handling at page level :

1. A JSP page can specify its own default error JSP page from an exception that is occurring within it, through the JSP error tag.
2. This enables a JSP page to specify its own handling of an error.
3. A JSP page that does not contain a JSP error tag has an error fall through to the application-level error JSP page.

   For example :

   i. Create a single error JSP page that handles the errors that occur across all the other JSP pages in the application. To specify a JSP page as an errorHandler page, use this JSP page directive :

   <%@ page isErrorPage="true" %>

   In the errorHandler JSP page, use ErrorDataBean or StoreErrorDataBean to retrieve more information about the exception and display messages.

   ii. Include the errorHandler JSP page in other JSP pages, by using this JSP directive to specify that if exceptions occur on the current page, forward the request to errorHandler.jsp :

   <%@ page errorPage="/errorHandler.jsp" %>

JSP directives :

1. Directives are JSP elements that provide global information about an entire JSP page.
2. All directives have scope of the entire JSP file.
3. The directive elements specify information about the page that remains the same between requests.

There are three possible directives currently defined by the JSP specification :

a. **Page directive :**

1. The page directive defines information that is globally available for JSP.
2. The page directive is a JSP tag that is used in almost every JSP file and defines a number of attributes that can affect the whole page.
3. The syntax is as follows :

<%@ page attribute%>

Example :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@page
import="com.javacode.examples.jspdirectivesexample.Pizza" %>
<%@ page import="java.util.*"%>
```

In the first line of example, language, contentType and pageEncoding page directive attributes are in the same directive statement.

b. **Include directive :**

1. The include directive is used to insert text and code at JSP translation time.
2. It includes a static file in a JSP file.
3. It has the following syntax :

<%@ include file = "relativeURL"%>

Example :

In the example "header.html" page is inserted into the Pizza form :

```
...
<head>
<meta charset="UTF-8">
<title>Jsp Directives Example</title>
<link rel="stylesheet" href="./static/css/pizzaorder.css">
</head>
<body>
<%@ include file="header.html" %>
<form action="orderResult.jsp" method="POST">
<h3>Pizza Types</h3>
<div>
...
```

header.html:

```
<h4>Java Code Examples<h4>
```

c. **Taglib directive :**

1. The taglib directive declares that the page uses custom user defined tags, it also defines the tags library.
2. The term custom tag refers to both tags and elements.
3. A tag is simply a short piece of a markup that is a part of JSP element.
4. The syntax is as follows :

<%@ taglib uri = "tagLibraryURI" prefix = "tagPrefix"%>

Example :
Following example shows a declaration sample in a JSP page :

```
<%@ taglib prefix="jgc" uri="WEB-INF/custom.tld"%>
...
<jgc:HelloWorld/>
...
```

Que 5.28. Explain various types of JSP scripting elements with example.

Answer

1. JSP scripting elements allows us to insert Java code into Java Servlet generated from JSP page.
2. Following are the four scripting elements :

a. JSP comment :

i. JSP comment is to document the code.

ii. JSP comment is used to note some parts of JSP page to make it clearer and easier to maintain.

iii. The syntax of JSP comment is as follows :

```
<%-- This is a JSP comment --%>
```

b. Expression :

i. The expression is one of the most basic scripting elements in JSP.

ii. The expression is used to insert value directly to the output.

iii. The syntax of an expression is as follows :

```
<%= expression%>
```

iv. It is noticed that there is no space between <% and =.

c. Scriptlet tag :

i. Scriptlet is similar to expression except for the equal sign "=".

ii. We can insert any plain Java code inside a scriptlet.

iii. The syntax of scriptlet tag is as follows :

```
<% // any java source code here %>
```

d. Declaration tag :

i. We can declare static member, instance variable and methods inside declaration tag.

ii. Syntax of declaration tag :

```
<%! declaration %>
```

Example :

```
<html>
<head>
```

```
<title>My First JSP Page</title>
</head>
<%--This is declaration tag--%>
<%!
int count = 0;
%>
<body>
Page Count is :
<% out.println(++count); %>
<%= new Java.util.Date()%>
</body>
</html>
```

Que 5.29. **Explain the difference between JSP includes directive elements and JSP includes action elements.**

Answer

S. No.	JSP include directive element	JSP include action element
1.	Resources are included at translation time.	Resources are included at request time.
2.	It uses file attribute to specify the resource.	It uses page attribute to specify the resource.
3.	Used to include static resource such as HTML.	Used to include dynamic resource such as JSP.
4.	It does not allow to pass parameters.	It allow to pass parameter using <jsp:param> action tag.
5.	It does not pass request and response object.	It allows us to pass request and response object.
6.	It can use both absolute and relative path to include resource.	It can use only relative path to include resource.

Que 5.30. **What are Java Beans ? Why they are used ? Write a JSP page and use an existing Java Beans in JSP page by using the standard action. Write the program with describing the output ?**

AKTU 2017-18, Marks 10

Answer

Java Beans and Java Beans are used because : Refer Q. 4.4, Page 4–5D, Unit-4.

JSP page :

Login.jsp :

```
<body>
<h2>Using Java Beans with JSP</h2>
<form method ="get" action = "http://localhost:7001/examplesWebApp/
Receive.jsp">
Enter User Name <input type="text" name="user"> <br>
Enter Password <input type="password" name="pass"> <br>
<input type="submit">
</form>
</body>
```

Receive.jsp :

```
<body>
<jsp:useBean id="snr" class="pack.ValidateBean" />
<jsp:setProperty name="snr" property="user" />
<jsp:setProperty name="snr" property="pass" />
You entered user name as <jsp:getProperty name="snr" property="user" />
<br>
You entered user password as <jsp:getProperty name="snr" property="pass"
/> <br>
<br>
You are a <%= snr.validate("Rao", "java") %> user. <br>
<b>Thank You</b>
</body>
```

ValidateBean.jsp :

```
package pack;
public class ValidateBean
{
String user;
String pass;
public ValidateBean( ) { }
public void setUser(String user)
{
this.user = user;
}
public String getUser( )
{
```

```
return user;
}
public void setPass(String pass)
{
this.pass = pass;
}
public String getPass( )
{
return pass;
}
public String validate(String s1,String s2)
{
if(s1.equals(user) && s2.equals(pass))
return "VALID";
else
return "INVALID";
}
}
```

Output :

Using Java Beans with JSP

Enter User Name [Rao]

Enter Password [****]

[Submit]

You entered user name as Rao
You entered user password as java
You are a VALID user.
Thank You

Explanation of Output :

1. The getProperty action calls get method and gets the value of the property.
2. The first statement calls getUser() method and retrieves the value of the variable user (set earlier with set method) and directly puts in the output stream of client.
   `<%= snr.validate("Rao", "java") %>`
3. The statement calls validate() method of ValidateBean and checks the user name and password entered by the user with Rao and java. The result of validation is returned to JSP expression which sends to client.

Que 5.31. Explain custom tag libraries with its declaration.

Answer

1. A custom tag library is a collection of the Tag Library Descriptor (TLD) and all files for a related set of custom actions.

2. TLD and all files are packaged in a JAR file.
3. A custom actions can access to all information about the request and can add content to the response body.
4. Custom action is implemented as a Java class or as a tag file.
5. The name of the tag file and other information are specified in a file called Tag Library Descriptor (TLD).

Declaration of custom tag library includes :

a. uri :
   i. The uri attribute find the class or tag file for each custom actions.
   ii. It contains a string container which is used to locate the TLD for the library.

b. prefix :
   i. The prefix attribute assigns a label to the tag library.
   ii. We use this label to reference its associated tag library when writing our pages using custom JSP tags.
   iii. Custom tag library defines a default prefix.

<mytaglib:newtag>

For example :

```
<%@ page contentType = "text/html" %>
<%@ taglib prefix = "ora" uri = "orataglib" %>   //This is custom tag library
<html>
<head>
<title>Messages of the Day </title>
</head>
<body bgcolor = "white">
<h1>Messages of the Day</h1>
<h2>Deep Thoughts - by Mahatma Gandhi</h2>
<i> <ora : motd category = "thoughts" /> </i>
<h2>Quotes From the Famous and the Unknown</h2>
<i> <ora : motd category = "quotes" /> </i>
</body>
</html>
```

Que 5.32. How data can be shared between JSPs ? Explain.

Answer

1. JSP application consists of more than a single page, and multiple pages often need access to the same information and server-side resources.
2. We need a way to pass data from one page to another when same request is processed by multiple pages.

Sharing session data :

1. When a user need same data over the multiple request in a session.
2. This type of sharing is for different requests of same user.
3. This type of sharing of data is called sharing of session data.

4. For example, in a travel agency, date and destination is important to remember for booking the flight.

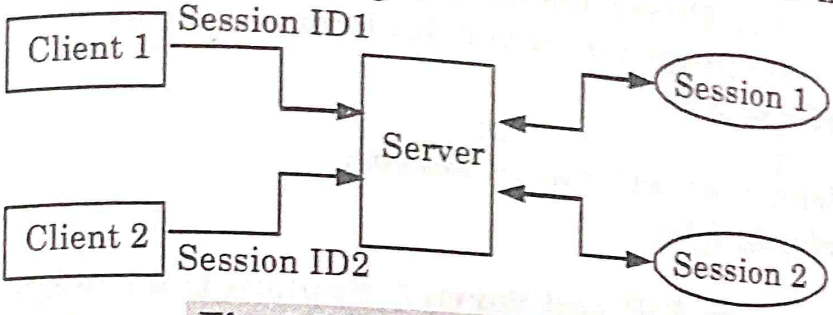


Fig. 5.32.1. Session scope.

Sharing application data :

1. When the different users make same request through multiple pages to an application.
2. In this case, the information is saved in the application by one page and later it can be accessed by another page, even if the two pages were requested by different users.
3. This type of sharing of data is called sharing of application data.

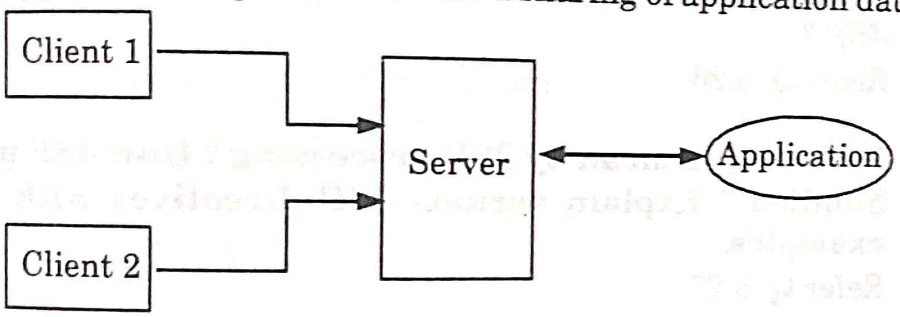


Fig. 5.32.2. Application scope.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as in UNIVERSITY EXAMINATION.

Q. 1. Explain the life cycle of servlet. Also write a servlet for displaying a string "HELLO WORLD!"

Ans. Refer Q. 5.2.

Q. 2. What is servlet ? Explain its life cycle. Illustrate some characteristics of servlet. How does servlet score over CGI ?

Ans. Refer Q. 5.4.

Q. 3. Explain how servlet works.

Ans. Refer Q. 5.7.