Unit V: Servlets and JSP

Servlets

- Servlet overview and architecture
- Servlet Interface and Servlet Life Cycle
- Handling HTTP post request
- Handling HTTP get request
- Redirecting request to other resources
- Session Tracking
- Session Tracking with HTTP Session
- Cookies

Java Server Pages (JSP)

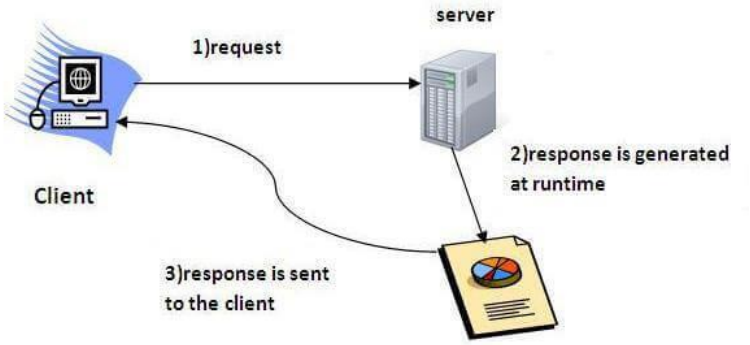
- Introduction
- Overview of JSP
- Scripting
- Directives
- Implicit Objects
- A first JSP example
- Standard actions
- Custom Tag Libraries

What is Servlet?

Servlet is a server side Java Technology or API used to create web applications. It provides a set of classes and interfaces required to advanced web applications under **javax.servlet** and **javax.servlet.http** packages. Some import classes and interfaces are

- Servlet interface
- GenericServlet class
- ServletRequest class
- ServletResponse class
- ServletException class
- HttpServlet class
- HttpServletRequest class
- HttpServletResponse class
- HttpSession interface

Java Servlet technology is used to receive a request from some web client, perform some required operations and returns the HTML contents to the web client.



A class to be called as servlet must be inherited from Servlet interface or its child class GenericServlet or HttpServlet class.

Servlet → GenericServlet → HttpServlet

Servlet interface and **GenericServlet** class provides a method called **service()** that we need to override to handle the request send by the client using get or post method and send response to the client. To get the data send the client as part of request is received using special class called **ServletRequest** and response is send back using **ServletResponse** class. These classes are passed as argument to the **service()** method. It throws **ServletException** if some runtime error occurs.

public void service(ServletRequest request, ServletResponse response) throws ServletException

ServletRequest class provides the methods to collect the data from client

- String getParameter(String parametername)

ServletResponse class is used to send the response to the client and redirect the control to other resource

- PrintWriter getWriter()
- void setContentType(String typename)
- void sendRedirect(String url)

Use **@WebServlet** annotation to define the **end point** to the servlet to be called by web client.

Example

Create a servlet that sends a welcome message to the web client

```
import javax.servlet.*;
import javax.servlet.annotation.*;
import java.io.*;

@WebServlet(name = "WelcomeServlet", value = "/WelcomeServlet")
public class WelcomeServlet extends GenericServlet {
    @Override
    public void service(ServletRequest request, ServletResponse response)
throws ServletException,IOException{
            PrintWriter out=response.getWriter();
            response.setContentType("text/html");
            out.println("<h1>Welcome to Servlet</h1>");
```

```
    }
}
```

What is doGet() and doPost() methods?

They are special methods provided in HttpServlet class to handle the request send by the client using post method or get method. We need to override these methods in our servlet class.

In case of **post** method, we must have some **HTML Form** to submit the data while in case of get method we can also send the data using **query string**.

Syntax

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException

Example 1

Create an HTML page (factorial.html) to input a number and send that number to a servlet (FactorialServlet.java) using post method. Show the factorial of that number.

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <form action="/FactorialServlet" method="post">
      Number <input type="text" name="num"><br>
      <input type="submit" value="Show Factorial">
  </form>
</body>
</html>
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;
@WebServlet(name = "FactorialServlet", value = "/FactorialServlet")
public class FactorialServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();
        int num=Integer.parseInt(request.getParameter("num"));
        int f=1;
        for(int i=1;i<num;i++) f=f*i;
        out.println("<h1>Factorial is "+f+"</h1>");
    }
}
```

Example 2

Create a servlet FactorialServlet.java using query string method. Show the factorial of that number.

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
```

```
import java.io.*;
@WebServlet(name = "FactorialServlet", value = "/FactorialServlet")
public class FactorialServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();
        int num=Integer.parseInt(request.getParameter("num"));
        int f=1;
        for(int i=1;i<num;i++) f=f*i;
        out.println("<h1>Factorial is "+f+"</h1>");
    }
}
```

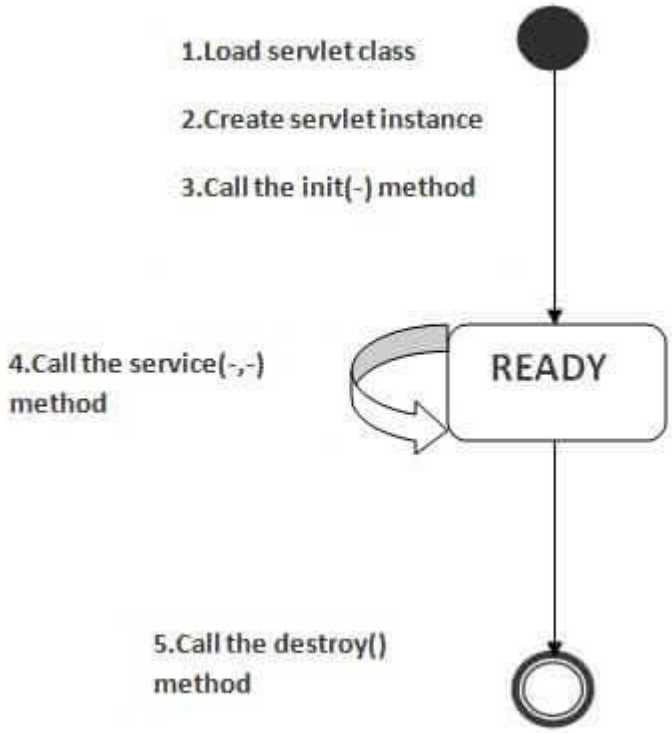
Execute the Servlet using the query string following URL

https://localhost:8080/FactorialServlet?num=7

Servlet Life Cycle

The servlet runs inside special container called as servlet container. Every servlet has its life cycle.

- Servlet class is loaded into memory
- Servlet instance get created
- init() method is invoked
- service() method is invoked
- destroy() method is invoked



A servlet has three states

- new
- ready
- end

The servlet is in new state when the servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the servlet container invokes the destroy() method, it shifts to the end state.

Syntax of methods

public void init(ServletConfig config) throws ServletException

public void service(ServletRequest request, ServletResponse response) throws ServletException

public void destroy()

How to send request to other resource?

To transfer the request to some other resource URL we need to use sendRedirect() method of ServletResponse or HttpResponse class.

Syntax

response.sendRedirect(String url);

Example

response.sendRedirect("dashboard");

What is session management or state management?

When we start a website and close a website, that duration is called as session. Sometimes we need to manage some information unique to the user that must be accessible among multiple servlets during the session. It is known as **session management** or **state management** or **session tracking**. E.g. we need to share your userid of current user among the multiple servlets after validation process is done.

There are various session tacking techniques

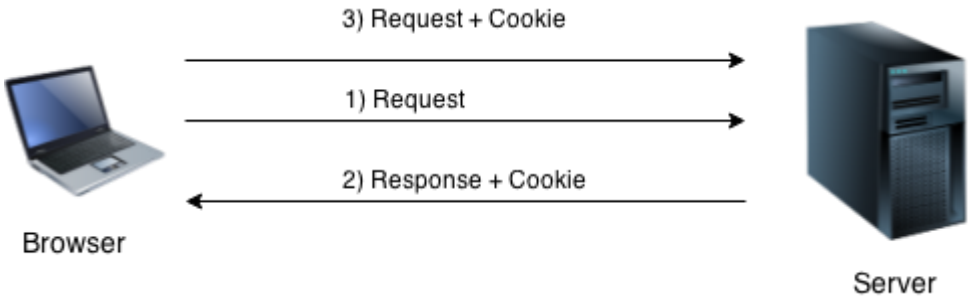
- HttpSession interface
- Cookies
- Hidden Form Field
- URL Rewriting

What are cookies in servlet?

A cookie is a small piece of information that is stored on local machine and accessible between the multiple client requests among all the servlets of a website.

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.

To add a cookie to the client machine, we need to add it with response. It returns to the server automatically with the request.



Cookies can be of two types

- Non-persistent cookie
- Persistent cookie

It is valid for single session only. It is removed each time when user closes the browser.

Persistent cookie is available for multiple session. It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

- It is simplest technique of maintaining the user state
- Cookies are maintained at client side, require less resources at the server side

Disadvantage of Cookies

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

How to create and use the cookies in Java Servlet?

Java Servlet provides **Cookie** class under **javax.servlet.http** package to create and use the cookies

- Cookie()
- Cookie(String cookiename, String cookievalue)
- public void setMaxAage(int durationInSeconds)
- public void setName(String cookiename)
- public String getName()
- public void setValue(String cookievalue)
- public String getValue()

Once a cookie is created, we need to add it to the **HttpResponse type object** using **addCookie()** method.

Example

```
Cookie c=new Cookie("user","demouser");//creating cookie object
response.addCookie(c);//adding cookie in the response
```

To read the cookie information, we need to use **HttpServletRequest type object** using **getCookies()** method

Example

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++)
  out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
```

What is HttpSession?

HttpSession is used to manage user specific information at the server side inside servlet container. It require more resources at the server side.

We need to get reference of session object using **getSession()** method of **HttpServletRequest type object** with two syntaxes

- public HttpSession getSession()
- public HttpSession getSession(boolean create)

In first method, the reference is returned if the session object is available inside the servlet container and NULL if session object is not available.

In second method, it creates a new session object if object is not available else returns existing object.

Use **setAttribute()** method of **HttpSession** interface to add the user state information to the HttpSession and **getAttribute()** method to read the value of the attribute in HttpSession.

Syntax

void setAttribute(String name, Object value)

Object getAttribute(String name)

Example 1: Creating a session variable

```
HttpSession session=request.getSession();
session.setAttribute("userid",userid);
```

Example 2: Reading the value of session variable

```
HttpSession session=request.getSession();
String userid=(String) session.getAttribute("userid");
```

Introduction to Java Server Pages (JSP)

JSP is a server side technology to build dynamic web pages by merging Java code into HTML at server side. It is a text document with .jsp file extension. It get compiled automatically by the Java Web Server (e.g. Tomcat, Glassfish, Web Logic, Web Sphare) into corresponding servlet class using special container called as JSP Container.

A JSP pages can have various elements

- HTML Tags
- Scriptlets
- Intrinsic objects
- Expressions
- JSP Tags or JSP Actions
- Directives
- Declaration

What are intrinsic objects?

Special built-in objects which are provided by JSP that are available to use and we do not need to create them are called as intrinsic objects. Some of them are

- request object (reference of HttpServletRequest class)
- response object (reference of HttpServletResponse class)
- out object (reference of PrintWriter class)
- session object (reference of HttpSession interface)
- config object
- page object
- application object
- exception object
- pageContext object

What is scriptlets?

Wherever we merge the Java code into HTML in JSP page we need to special delimiters <% and %>. Such code is known as scriptlet.

Example: first.jsp

```
<h1>First JSP Page</h1>
<%
    out.println("Welcome to JSP");
%>
```

Hands-on Lab

- XAMPP Server
- Place JSP Pages inside the folder c:\xampp\tomcat\webapps\ROOT
- Run the web page using URL http://localhost:8080/first.jsp

What is expression?

The expressions are used to merge the output inside the HTML code without need of out.prinln() method. Use <%= and %> delimiters to define the expressions.

Example: expressiontest.jsp

```
<h1>First JSP Page</h1>
<p>Message is <%="Welcome to JSP"%></p>
<p>Message is <%=3+4*3%></p>
```

How to read the request data in JSP?

- Use **getParameter()** method of request object to read the data

Syntax

String request.getParameter(String paramname)

Example

String name=request.getParameter("name");

How to send control to some other resource URL?

- Use **sendRedirect()** method of response object to send the control to other resource URL

Syntax

response.sendRedirect(String url)

Example

response.sendRedirect("dashboard.jsp");

What is declaration?

- Declaration is used to define global variables and the methods.
- Use <%! and %> delimiters to user the declaration

Example: declaration.jsp

```
<%!
   public int num=5;
   public int factorial(int n){
      if(n==1)
         return 1;
      else
         return n*factorial(n-1);
   }
```

```
%>
<p>Value of num is <%=num%></p>
<p>Factorial of 6 is <%=factorial(6)%></p>
```

What are the directives?

Special instructions given to the compiler are called as directives. JSP provides <%@ and %> delimiters to define the directives. Some of the directives are

- page directive
- include directive
- taglib directive

Syntax to use the directive

<%@ directive attribute="value" %>

Important Attributes of page directive are

- import
- contentType
- extends
- isThreadSafe
- autoFlush
- session
- errorPage
- isErrorPage

The **import** attribute is used to import one or more classes of a package.

The **contentType** attribute is used to define the type of contents to be send to the client

Example

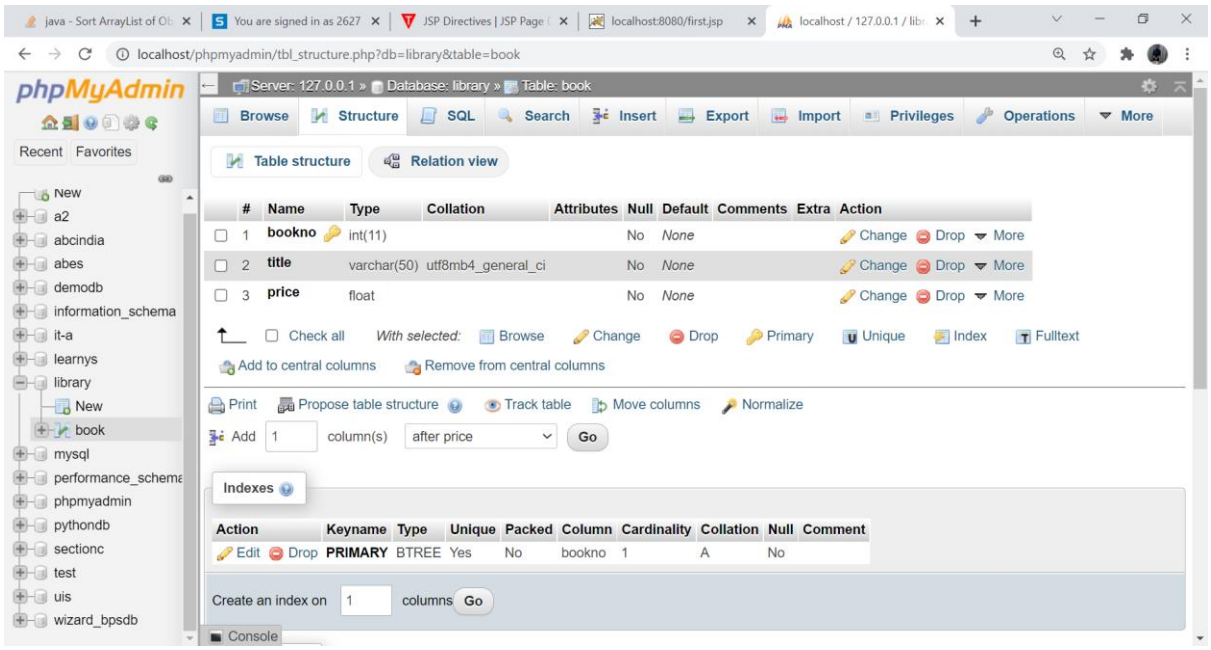
<%@ page import="java.sql.*" contentType="text/html"%>

Test Case: Using JDBC with JSP

Create a database **library** using MySQL RDBMS placed on **localhost** at port number **3306** having a table **book** with columns bookno (int), title (varchar) and price (float).

Create a web page to input the data of books and save into database table using JSP Technology and JDBC API.

Database and Table Structure



HTML Page Contents (book.html)

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <h1>Book Information</h1>
        <form action="savebook.jsp" method="post">
            Book No <input type="text" name="bookno" id="bookno"><br>
            Title <input type="text" name="title" id="title"><br>
            Price <input type="text" name="price" id="price"><br>
            <input type="submit">
        </form>
    </body>
</html>
```

JSP Page contents (savebook.jsp)

```
<%@page import="java.sql.*"%>
<%
    int bookno=Integer.parseInt(request.getParameter("bookno"));
    String title=request.getParameter("title");
    float price=Float.parseFloat(request.getParameter("price"));
    try{
        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
        Connection
cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root",""
);
        String sql="INSERT INTO book VALUES(?,?,?)";
        PreparedStatement ps=cn.prepareStatement(sql);
```

```
        ps.setInt(1,bookno);
        ps.setString(2,title);
        ps.setFloat(3,price);
        ps.executeUpdate();
        cn.close();
        out.println("Record Saved");
    }catch(Exception ex){
        out.println("Error : "+ex.getMessage());
    }
%>
```

View all records using JSP and JDBC (readalldata.jsp)

```
<%@ page import="java.sql.*"%>
<h1>Data of Books in the libary</h1>
<%
    String sql="SELECT * FROM book";
    try{
        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
        Connection
cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root",""
);
        Statement st=cn.createStatement();
        ResultSet rs=st.executeQuery(sql);
        while(rs.next()){
            out.println(rs.getString(1)+","+rs.getString(2)+","+rs.getString(3
)+"<br>");
        }
        cn.close();
    }catch(Exception ex){
        out.println("Error : "+ex.getMessage());
    }
%>
```

What are JSP Actions or JSP Action Tags?

JSP also provides some predefined tags to perform some advance operations. All such tags are prefixed with jsp:

Some of the JSP action tags are

- <jsp:forward/>
  - forwards the request and response to another resource
- <jsp:include/>
  - includes some resource
- <jsp:param/>
  - sets the parameter value in forward and include
- <jsp:useBean/>
  - creates a bean object
- <jsp:setProperty/>

- sets the value of property in bean object

- <jsp:getProperty/>
  - returns the value of property of the bean

Using <jsp:forward/> action

- It is used to forward the request from one JSP to another JSP page
- It allows to add some parameters while forwarding
- It has two syntaxes

Syntax 1

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

Syntax 2

```
<jsp:forward page="url">
        <jsp:param name="parametername" value="parametervalue" />
</jsp:forward>
```

Using <jsp:include/> action

- Used to include a resource file in your JSP page
- It can be an HTML page, JSP page, Java Servlet or any other resource
- It allows to reuse a page in multiple JSP pages

It also has two syntaxes

Syntax 1

<jsp:include page="filename" />

Syntax 2

<jsp:include page="filename">

<jsp:param name="parametername" value="parametervalue" />

</jsp:include>

Test Case: Use of <jsp:import/> and <jsp:forward/> directives

Create an HTML page as **footer.htm** having some copyright details.

Create a **signup.html** page to signup a new user with fields as **userid**, **password**, **username, usertype** and save the user data in a table **users** under **library** database using **saveuser.jsp** page

Create a login page as **login.jsp** to input user id and password. Include **footer.html** file at the bottom of the page. Verify the user from users table using **verifyuser.jsp** page. If usertype is admin then transfer the control to **admin.jsp** page else transfer the control to **general.jsp** page

Contents of footer.html file

```
<p align='center'>
Copyright @ ABES Engineering College, Ghaziabad, 2021
</p>
```

Contents of signup.html file

```
<form action="saveuser.jsp" method="post">
    User ID <input type="text" name="userid"><br>
    Password <input type="password" name="password"><br>
    User Name <input type="text" name="username"><br>
    User Type <select name="usertype">
        <option value="admin">Admin</option>
        <option value="general">General</option>
        </select><br>
    <input type="submit" value="Register">
</form>
```

Contents of saveuser.jsp file

```
<%@ page import="java.sql.*" %>
<%
    String userid=request.getParameter("userid");
    String password=request.getParameter("password");
    String username=request.getParameter("username");
    String usertype=request.getParameter("usertype");

    try{
        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
        Connection
cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root",""
);
        String sql="INSERT INTO users VALUES(?,?,?,?)";
        PreparedStatement ps=cn.prepareStatement(sql);
        ps.setString(1,userid);
        ps.setString(2,password);
        ps.setString(3,username);
        ps.setString(4,usertype);
        ps.executeUpdate();
        cn.close();
        out.println("User Created");
    }catch(Exception ex){
        out.println("Error : "+ex.getMessage());
    }

%>
```

Contents of login.jsp file

```
<form action="verifyuser.jsp" method="post">
    User ID : <input type="text" name="userid"><br>
    Password : <input type="password" name="password"><br>
    <input type="submit" value="Login">
</form>

<jsp:include page="footer.html"/>
```

Contents of verifyuser.jsp using JSP Actions

```
<%@ page import="java.sql.*"%>
<%
    String userid=request.getParameter("userid");
    String password=request.getParameter("password");
    try{
        String sql="SELECT * FROM users WHERE userid=? and password=?";
        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
        Connection
cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root",""
);
        PreparedStatement ps=cn.prepareStatement(sql);
```

```
            ps.setString(1,userid);
            ps.setString(2,password);
            ResultSet rs=ps.executeQuery();
            if(rs.next()){
                String usertype=rs.getString("usertype");
                if(usertype.equals("admin")){
%>
                    <jsp:forward page="admin.jsp"/>
<%
                }
                else{
%>
                    <jsp:forward page="general.jsp"/>
<%
                }
            }
            else{
                out.println("Sorry! User Not found");
            }
            cn.close();

        }catch(Exception ex){
            out.println("Error : "+ex.getMessage());
        }
%>
```

What is RequestDispatcher?

The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.

It can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

Methods of RequestDispatcher interface

- public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException
    - Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException
    - Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

How to get the object of RequestDispatcher?

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher.

Syntax:

public RequestDispatcher getRequestDispatcher(String resource);

Examples

RequestDispatcher rd=request.getRequestDispatcher("admin.jsp");

rd.forward(request, response);

RequestDispatcher rd=request.getRequestDispatcher("footer.html");

rd.include(request, response);

Contents of verifyuser.jsp using RequestDispatcher

```
<%@ page import="java.sql.*"%>
<%
    String userid=request.getParameter("userid");
    String password=request.getParameter("password");
    try{
        String sql="SELECT * FROM users WHERE userid=? and password=?";
        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
        Connection
cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root",""
);
        PreparedStatement ps=cn.prepareStatement(sql);
        ps.setString(1,userid);
        ps.setString(2,password);
        ResultSet rs=ps.executeQuery();
        if(rs.next()){
            String usertype=rs.getString("usertype");
            if(usertype.equals("admin")){
                RequestDispatcher
rd=request.getRequestDispatcher("admin.jsp");
                rd.forward(request, response);
            }
            else{
                RequestDispatcher
rd=request.getRequestDispatcher("general.jsp");
                rd.forward(request, response);
            }
        }
        else{
            out.println("Sorry! User Not found");
        }
        cn.close();

    }catch(Exception ex){
        out.println("Error : "+ex.getMessage());
    }
%>
```

What is Java Bean?

A JavaBean is a reusable Java class having data members, constructors, setters and getters. Instances of such classes can expose themselves to other Java Beans.

Special software called Bean Development Kit (BDK) is used test the bean components.

JSP can also use such classes to build dynamic web pages using JSP action tags

- <jsp:useBean/>
- <jsp:setProperty/>
- <jsp:setProperty/>

The jsp:useBean action tag is used to instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

Syntax of useBean

<jsp:useBean id= "instanceName" class= "className"/>

Syntax of setProperty

<jsp:setProperty name="instanceOfBean" property="propertyName" value="value"/>

Syntax of getProperty

<jsp:getProperty name="instanceOfBean" property="propertyName" />

University examination questions

- Compare doGet() and doPost() methods? (2020-21, 2 Marks)
- Write difference JSP and Servlet? (2020-21, 2 Marks)
- What is need of dynamic webpage? What are the advantages and issues involved in dynamic web page? (2020-21, 10 Marks)
- Write down the steps to connect database with web application using JDBC. (2020-21, 10 Marks)
- Explain Request Dispatcher. Also describe different ways to get the object of request dispatcher (2020-21, 10 Marks)
- Explain servlets with its life cycle. How its life cycle is different from life cycle of JSP? Explain with example (2020-21, 10 Marks)
- JSP is an extension of Servlet and not replacement. Justify? How problems of Servlet Technology can JSP supposed to solve? (2020-21, 10 Marks)
- Discuss about tomcat server. How to set the classpath for servlet in tomcat server. (2019-20, 2 Marks)

- Explain Servlets with its life cycle. How its life cycle is differ from life cycle of JSP? Explain with an example (2019-20, 10 Marks)
- Discuss JSP in detail. What are JSP directives? Explain various types of directives with suitable example. (2019-20, 10 Marks)
- Compare JSP and Servlet. Explain the life cycle of JSP page with suitable diagram. Also list any five action tags used in JSP. (2018-19, 7 Marks)
- What is the difference between session and cookies? Write a servlet program for login and logout. (2018-19, 7 Marks)
- What is web project? (2017-18, 2 Marks)
- Explain client-server technology with diagram. (2017-18, 2 Marks)
- What are JSP directives? Explain various types of directives with example? (2017-18,10 Marks)
- Explain implicit objects available in jsp with example? (2017-18,10 Marks)
- What are standard actions in JSP? Illustrate with example? (2017-18,10 Marks)
- Explain the Life cycle of servlet? Also write a servlet for displaying a string “HELLO WORLD!”. (2017-18,10 Marks)
- Create a Form in HTML taking Account Number from user as input then write a servlet program receiving this form 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. (2017-18,10 Marks)
- What are java Beans? Why they are used? Write a JSP page and use an existing java bean in JSP page by using the standard action. Write the program with describing the output? (2017-18,10 Marks)