

Session 2

JSP

JavaServer Pages put Java inside HTML so that the view is easier to write than a servlet full of `println` calls. The session covers scripting elements, JSTL, JDBC from a page, action elements, implicit objects and a small combined project.

Objectives

✗ Do not copy. Read for understanding and the viva

- Complete questions 6 to 12 of the manual: jsp
- Prepare the deliverable before the lab and finish it during the session
- Be ready to explain every step in the viva

Questions Covered

✗ Do not copy. Read for understanding and the viva

Question	Requirement	Status
Q6	Write JSP Programme to print current date and time along with timestamp, implement...	Complete
Q7	Create a JSP page and implement a Scripting Tag, Expression tag and Declaration tag	Complete
Q8	Import JSTL library in JSP Page and use its following tags	Complete
Q9	Create a JSP Page for database connectivity using JDBC and show the students details...	Complete
Q10	Write a JSP application using following Action Elements	Complete
Q11	Write a JSP program using the following implicit objects with an example	Complete
Q12	Create a JSP Project implementing all the above (Session 1 and Session 2) concepts...	Complete

Preparation

✗ Do not copy. Read for understanding and the viva

- Add the JSTL jars (jakarta.servlet.jsp.jstl and its API) to the project; JSTL is not part of Tomcat.
- For auto-refresh use `response.setHeader("Refresh", "5")` or a meta refresh tag.
- Question 12 is a mini project: plan its pages (login, list, add, edit, delete, error) before coding.
- One Maven web project `JspLab` holds all seven answers, with the same `pom.xml` as Session 1. JSP pages go in `src/main/webapp/`, Java classes in `src/main/java/ignou/`. Questions 9 and 12 need the IGNOU database from Session 1, Question 5.

Project Setup

✗ Do not copy. Read for understanding and the viva

1. Create the project as in Session 1 (NetBeans: File, New Project, Java with Maven, Web Application, name, Next, server Tomcat 10.1, Finish; Eclipse: Dynamic Web Project, then Convert to Maven Project) and name it `JspLab`. Copy the Session 1 `pom.xml`, changing `artifactId` and `finalName`.
2. To add a page: right-click Web Pages (NetBeans) or `src/main/webapp` (Eclipse), New, JSP, name without extension. Base URL after Run: `http://localhost:8080/JspLab/`.

Question 6

Problem Statement

■ Write in lab record

Write JSP Programme to print current date and time along with timestamp, implement auto-refresh of a page.

Solution

■ Write in lab record

Steps

1. New JSP `datetime` in `src/main/webapp/`; paste the listing.
2. Run and open `http://localhost:8080/JspLab/datetime.jsp`. Leave it for 20 seconds: the seconds and the timestamp advance on their own every 5 seconds.
3. In the browser dev tools, Network tab, watch a new request appear every 5 seconds.

Program

datetime.jsp

```
1  <%-- Q6: src/main/webapp/datetime.jsp
2      Prints date, time and timestamp; the page reloads itself every 5 second
3  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4  <%@ page import="java.util.Date, java.text.SimpleDateFormat, java.sql.Timest
5  <%
6      // HTTP Refresh header: the browser requests this URL again after 5 seco
7      response.setHeader("Refresh", "5");
8      long millis = System.currentTimeMillis();
9      Date now = new Date(millis);
10 %>
11 <!DOCTYPE html>
12 <html>
13 <head>
14     <meta charset="UTF-8">
15     <%-- Second way: meta refresh. Either one alone is enough. --%>
16     <meta http-equiv="refresh" content="5">
17     <title>Date and Time (auto refresh)</title>
18 </head>
19 <body>
20     <h2>Current Date and Time</h2>
21     <p>Date: <%= new SimpleDateFormat("dd-MM-yyyy").format(now) %></p>
22     <p>Time: <%= new SimpleDateFormat("HH:mm:ss").format(now) %></p>
23     <p>Full: <%= now %></p>
24     <p>Timestamp (ms since epoch): <%= millis %></p>
25     <p>SQL Timestamp: <%= new Timestamp(millis) %></p>
26     <p><i>This page refreshes every 5 seconds. Watch the seconds change.</i>
27 </body>
28 </html>
```

Output

Checked by reading, not executed (no Tomcat here). The page repaints itself every five seconds; one snapshot:

```
1 Current Date and Time
2 Date: 26-09-2026
3 Time: 11:02:15
4 Full: Sat Sep 26 11:02:15 IST 2026
5 Timestamp (ms since epoch): 1790398335518
6 SQL Timestamp: 2026-09-26 11:02:15.518
7 This page refreshes every 5 seconds. Watch the seconds change.
```

Explanation

- Tomcat compiles the JSP into a servlet on first request
(`work/Catalina/localhost/JspLab/org/apache/jsp/datetime_jsp.java`); HTML becomes `out.write` calls, scriptlet code is copied into `_jspService` .
- `response.setHeader("Refresh", "5")` asks the browser to re-request the URL after 5 seconds; the `meta http-equiv="refresh"` tag does the same from the HTML side. One is enough; both are shown so you can explain either in the viva. Headers must be set before output is committed, so the scriptlet sits above the DOCTYPE.
- `System.currentTimeMillis()` is the timestamp; `SimpleDateFormat` turns it into date and time strings.

Question 7

Problem Statement

■ Write in lab record

Create a JSP page and implement a Scripting Tag, Expression tag and Declaration tag.

Solution

■ Write in lab record

Steps

1. New JSP `scripting` in `src/main/webapp/` .
2. Open `http://localhost:8080/JspLab/scripting.jsp` , reload several times, then open `scripting.jsp?name=Asha` .

Program

scripting.jsp

```
1 <!-- Q7: src/main/webapp/scripting.jsp
2     Declaration <%! %>, scriptlet <% %> and expression <%= %> in one page.
3 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4
5 <!-- Declaration: becomes a field and a method of the generated servlet class
6     Because it is a field, hitCount survives across requests (until redeploy)
7 <%!
8     private int hitCount = 0;
9
10    private long factorial(int n) {
11        long f = 1;
12        for (int i = 2; i ≤ n; i++) f *= i;
13        return f;
14    }
15 %>
16
17 <!-- Scriptlet: plain Java inside _jspService(); runs on every request. --%>
18 <%
19     hitCount++;
20     String name = request.getParameter("name");
21     if (name == null || name.isBlank()) name = "Student";
22     int n = 5;
23 %>
24 <!DOCTYPE html>
25 <html>
26 <head><meta charset="UTF-8"><title>JSP Scripting Elements</title></head>
27 <body>
28     <h2>JSP Scripting Elements</h2>
29     <!-- Expression: value is converted to String and written into the output
30     <p>Hello, <%= name %>! You are visitor number <%= hitCount %>.</p>
31     <p>Factorial of <%= n %> is <%= factorial(n) %>.</p>
32
33     <h3>Multiplication table of 7 (scriptlet loop)</h3>
34     <table border="1" cellpadding="4">
35         <% for (int i = 1; i ≤ 10; i++) { %>
36             <tr><td>7 x <%= i %></td><td> <%= 7 * i %></td></tr>
37         <% } %>
38     </table>
39
40     <form method="get">
41         <p>Your name: <input name="name" value="<%= name %>"> <button type="
42     </form>
```

```

43 </body>
44 </html>

```

Output

Checked by reading. Third visit with `?name=Asha` :

```

1 JSP Scripting Elements
2 Hello, Asha! You are visitor number 3.
3 Factorial of 5 is 120.
4 Multiplication table of 7 (scriptlet loop)
5 7 x 1 = 7 ... 7 x 10 = 70 (ten rows)
6 Your name: [Asha] [Greet]

```

Explanation

Element	Syntax	Becomes in the generated servlet	Used here for
Declaration	<code><%! ... %></code>	Class-level field or method, outside <code>_jspService</code>	<code>hitCount</code> field, <code>factorial()</code> method
Scriptlet	<code><% ... %></code>	Statements inside <code>_jspService</code> , run on every request	Reading <code>name</code> , incrementing <code>hitCount</code> , the <code>for</code> loop
Expression	<code><%= ... %></code>	<code>out.print(...)</code> of the value	Printing <code>name</code> , <code>hitCount</code> , <code>factorial(n)</code> , <code>7 * i</code>

- `hitCount` keeps counting across requests because it is a field of the one servlet instance; it resets when the page is recompiled or the application redeploys. It is also shared by all users and not thread-safe, which is why real applications keep such state in a session or a database.
- An expression must not end with a semicolon; a scriptlet must.
- The loop opens in one scriptlet and closes in another; the HTML row between them is emitted on every iteration.

Question 8

Problem Statement

■ Write in lab record

Import JSTL library in JSP Page and use its following tags:

1. out
2. if
3. forEach
4. choice, when and otherwise
5. url and redirect

Solution

■ Write in lab record

Steps

1. Confirm the two JSTL artifacts (`jakarta.servlet.jsp.jstl-api` 3.0.0 and Glassfish `jakarta.servlet.jsp.jstl` 3.0.1) are in `pom.xml` ; without Maven, download both jars and drop them into `src/main/webapp/WEB-INF/lib/` .
2. New JSP `jstl-demo` ; keep `datetime.jsp` from Question 6 as the redirect target.
3. Open `http://localhost:8080/JspLab/jstl-demo.jsp` , then click the two links at the bottom.

Program

jstl-demo.jsp

```

1 <%-- Q8: src/main/webapp/jstl-demo.jsp
2     JSTL 3.0 core tags on Tomcat 10.1: the taglib URI is jakarta.tags.core
3 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4 <%@ taglib prefix="c" uri="jakarta.tags.core" %>
5
6 <%-- 5. c:redirect: /jstl-demo.jsp?go=clock sends the browser to datetime.js
7 <c:if test="${param.go = 'clock'}">
8     <c:redirect url="/datetime.jsp" />
9 </c:if>
10
11 <%-- test data: a list of marks in page scope --%>
12 <c:set var="student" value="Asha Verma" />
13 <c:set var="marks" value="${[78, 92, 45, 66, 88]}" />
14 <!DOCTYPE html>
15 <html>
16 <head><meta charset="UTF-8"><title>JSTL Core Tags</title></head>
17 <body>
18     <h2>JSTL Core Tag Demo</h2>
19
20     <h3>1. c:out</h3>
21     <%-- escapeXml is true by default, so a value like <b>x</b> is printed l
22     <p>Student: <c:out value="${student}" /></p>
23     <p>Unknown parameter with default: <c:out value="${param.city}" default=
24     <p>Escaped: <c:out value="<b>bold?</b>" /></p>
25
26     <h3>2. c:if</h3>
27     <c:if test="${not empty param.name}">
28         <p>Hello, <c:out value="${param.name}" /></p>
29     </c:if>
30     <c:if test="${empty param.name}">
31         <p>Add ?name=YourName to the URL to see c:if fire.</p>
32     </c:if>
33
34     <h3>3. c:forEach</h3>
35     <table border="1" cellpadding="4">
36         <tr><th>#</th><th>Marks</th><th>Result</th></tr>
37         <c:forEach var="m" items="${marks}" varStatus="st">
38             <tr>
39                 <td>${st.count}</td>
40                 <td>${m}</td>
41                 <td>
42                     <%-- 4. c:choose / c:when / c:otherwise --%>
43                     <c:choose>

```

```
44         <c:when test="{m ≥ 75}">Distinction</c:when>
45         <c:when test="{m ≥ 50}">Pass</c:when>
46         <c:otherwise>Fail</c:otherwise>
47     </c:choose>
48 </td>
49 </tr>
50 </c:forEach>
51 </table>
52 <p>Counting with begin/end/step:
53     <c:forEach var="i" begin="1" end="10" step="3">{i} </c:forEach>
54 </p>
55
56 <h3>5. c:url and c:redirect</h3>
57 <!-- c:url prefixes the context path and adds the session id if cookies
58 <c:url var="selfLink" value="/jstl-demo.jsp">
59     <c:param name="name" value="Rahul Singh" />
60     <c:param name="city" value="Kolkata" />
61 </c:url>
62 <p><a href="{selfLink}">Reload with name and city (built by c:url)</a><
63 <p>Rendered link: <c:out value="{selfLink}" /></p>
64 <c:url var="clockLink" value="/jstl-demo.jsp"><c:param name="go" value="
65 <p><a href="{clockLink}">Go to the clock page (c:redirect)</a></p>
66 </body>
67 </html>
```

Output

Checked by reading. First visit without parameters:

```
1 JSTL Core Tag Demo
2 1. c:out
3 Student: Asha Verma
4 Unknown parameter with default: not given
5 Escaped: <b>bold?</b>
6 2. c:if
7 Add ?name=YourName to the URL to see c:if fire.
8 3. c:forEach
9 # Marks Result
10 1 78 Distinction
11 2 92 Distinction
12 3 45 Fail
13 4 66 Pass
14 5 88 Distinction (five rows)
15 Counting with begin/end/step: 1 4 7 10
16 5. c:url and c:redirect
17 Reload with name and city (built by c:url)
18 Rendered link: /JspLab/jstl-demo.jsp?name=Rahul+Singh&city=Kolkata
19 Go to the clock page (c:redirect)
```

Clicking the first link shows `Hello, Rahul Singh` under `c:if` and `Kolkata` under `c:out`. Clicking the second sends the browser to `/JspLab/datetime.jsp` with an HTTP 302 and the address bar changes.

Explanation

- The taglib directive `uri="jakarta.tags.core"` is the JSTL 3.0 name; the old `java.sun.com` URI belongs to JSTL 1.2 on `javax` and fails on Tomcat 10.1 with "cannot be resolved".
- `c:out` prints an EL value and escapes XML characters by default, so `<b>` shows literally; `default` covers missing values. `c:if` has no else; for branches use `c:choose` with `c:when` and one `c:otherwise` (the manual's "choice").
- `c:forEach` walks any collection or array (`items`) or counts (`begin` , `end` , `step`). `varStatus` gives `index` , `count` , `first` , `last` .
- `c:url` prepends the context path, URL-encodes nested `c:param` values and appends `;jsessionid` when cookies are off. `c:redirect` sends a 302 and stops the page; it must run before any output, which is why it sits at the top.

Question 9

Problem Statement

■ Write in lab record

Create a JSP Page for database connectivity using JDBC and show the students details from the database created during exercise no 5 in session 1.

Solution

■ Write in lab record

Steps

1. MySQL must be running with the IGNOU database from Session 1 (`schema.sql`).
2. `mysql-connector-j` is already in `pom.xml` ; if not using Maven, copy the jar to `WEB-INF/lib/` .
3. New JSP `student-list` ; open `http://localhost:8080/JspLab/student-list.jsp` . Stop MySQL and reload once to see the error branch.

Program

student-list.jsp

```
1 <!-- Q9: src/main/webapp/student-list.jsp
2     Reads the Student table of the IGNOU database (Session 1, Q5) with plai
3 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4 <%@ page import="java.sql.Connection, java.sql.DriverManager, java.sql.Prepa
5 <!DOCTYPE html>
6 <html>
7 <head><meta charset="UTF-8"><title>Student List (JDBC)</title></head>
8 <body>
9     <h2>Students in IGNOU database</h2>
10 <%
11     String url = "jdbc:mysql://localhost:3306/IGNOU?useSSL=false&serverTimez
12     String sql = "SELECT enrolment_no, name, dob, email, mobile, programme,
13         + "FROM Student ORDER BY enrolment_no";
14     int count = 0;
15     // try-with-resources closes ResultSet, PreparedStatement and Connection
16     try (Connection con = DriverManager.getConnection(url, "ignou", "ignou12
17         PreparedStatement ps = con.prepareStatement(sql);
18         ResultSet rs = ps.executeQuery()) {
19 %>
20     <table border="1" cellpadding="4">
21         <tr><th>Enrolment</th><th>Name</th><th>DOB</th><th>Email</th><th>Mob
22         <th>Programme</th><th>Semester</th><th>Courses</th></tr>
23 <%
24         while (rs.next()) {
25             count++;
26 %>
27         <tr>
28             <td><%= rs.getString("enrolment_no") %></td>
29             <td><%= rs.getString("name") %></td>
30             <td><%= rs.getDate("dob") %></td>
31             <td><%= rs.getString("email") %></td>
32             <td><%= rs.getString("mobile") %></td>
33             <td><%= rs.getString("programme") %></td>
34             <td><%= rs.getInt("semester") %></td>
35             <td><%= rs.getString("courses") %></td>
36         </tr>
37 <%
38     }
39 %>
40 </table>
41 <p>Total students: <%= count %></p>
42 <%
43     } catch (SQLException e) {
```

```

44 %>
45     <p style="color:red">Database error: <%= e.getMessage() %></p>
46     <p>Check that MySQL is running, schema.sql was executed and mysql-connec
47 <%
48     }
49 %>
50 </body>
51 </html>

```

Output

Checked by reading. With the three seeded rows:

```

1 Students in IGNOU database
2 Enrolment   Name           DOB           Email           Mobile          Pr
3 2451001234  Asha Verma    2001-03-14   asha.verma@example.com  9876543210  MC
4 2451001235  Rahul Singh   2000-11-02   rahul.singh@example.com  9123456780  MC
5 2451001236  Meera Nair    2002-07-25   meera.nair@example.com   9988776655  MC
6 Total students: 3

```

With MySQL stopped: `Database error: Communications link failure` in red, followed by the hint line.

Explanation

- The JDBC steps are the same as in the DAO of Session 1: `DriverManager.getConnection(url, user, password)`, `prepareStatement`, `executeQuery`, loop over `ResultSet`. Connector/J 8 registers its driver automatically through the service loader, so `Class.forName("com.mysql.cj.jdbc.Driver")` is optional.
- The `try` with resources declares the connection, statement and result set together; they close in reverse order whether the loop finishes or throws.
- The scriptlet is split around the HTML so the header is written once and a row per `rs.next()`. SQL inside a JSP is acceptable here but mixes data access with presentation; Question 12 moves it back into `StudentDao`.

Question 10

Problem Statement

■ Write in lab record

Write a JSP application using following Action Elements

1. jsp:forward
2. jsp:include
3. set and getProperty
4. jsp:useBean

Solution

■ Write in lab record

Steps

1. Add class `StudentBean` in package `ignou` (Source Packages, New, Java Class).
2. Add three JSPs in `src/main/webapp/`: `header`, `action-demo`, `bean-view`.
3. Open `http://localhost:8080/JspLab/action-demo.jsp`. Submit with a name: `bean-view.jsp` shows the bean. Submit with the name empty: you are forwarded back to the form with a red message while the address bar still says `bean-view.jsp`.

Program

■ Lab record: every tab is one file of the answer. Write all of them.

StudentBean.java

StudentBean.java

```
1 package ignou;
2
3 import java.io.Serializable;
4
5 /**
6  * Q10: JavaBean for jsp:useBean. Rules a bean must follow: public class,
7  * public no-argument constructor, private fields, public getters and setter
8  * named after the fields. jsp:setProperty property="*" matches request
9  * parameter names to these setter names.
10 */
11 public class StudentBean implements Serializable {
12     private String name;
13     private String programme;
14     private int semester;
15
16     public StudentBean() { }
17
18     public String getName() { return name; }
19     public void setName(String name) { this.name = name; }
20     public String getProgramme() { return programme; }
21     public void setProgramme(String programme) { this.programme = programme; }
22     public int getSemester() { return semester; }
23     public void setSemester(int semester) { this.semester = semester; }
24 }
```

Output

Checked by reading. Submitting name , programme , semester

```
1 IGNOU Web Technologies Lab | Page title: Bean View | Served at Sat Sep 26 11
2 Values read back with jsp:getProperty
3 Name      Meera Nair
4 Programme MCA
5 Semester  2
6 After setProperty with a literal value, programme = MCA (forced)
7 Back
```

Submitting with an empty name leaves the address bar at but renders the form page with the header titled and the red line

forwarded back by `jsp:forward()` .

Explanation

Action	What it does	Where in the code
<code>jsp:include</code>	Runs another resource at request time and inserts its output; <code>jsp:param</code> adds request parameters visible only inside it	<code>header.jsp</code> at the top of both pages, with <code>title</code>
<code>jsp:forward</code>	Hands the same request to another page and discards any output so far; the browser URL does not change	Empty name in <code>bean-view.jsp</code>
<code>jsp:useBean</code>	Looks for a bean with that <code>id</code> in the given scope, creates one with the no-arg constructor if absent	<code>student</code> in request scope
<code>jsp:setProperty</code>	Calls setters; <code>property="*" matches every request parameter to a same-named setter and converts types</code>	Fills name, programme, semester (String to int)
<code>jsp:getProperty</code>	Calls the getter and prints the value	The three table rows

- `jsp:include` differs from the `include` directive: the directive pastes source at translation time, the action calls the page at request time, so the header's date is always current. The forward happens before any output; forwarding after content was flushed throws `IllegalStateException` .

Question 11

Problem Statement

■ Write in lab record

Write a JSP program using the following implicit objects with an example:

1. out

2. request
3. response
4. session
5. pageContext
6. exception

Solution

■ Write in lab record

Steps

1. Add JSPs `implicit` and `error` in `src/main/webapp/`.
2. Open `http://localhost:8080/JspLab/implicit.jsp?name=Asha`, reload twice, then click the divide-by-zero link. In dev tools, Network, the response headers show `X-Lab: MCSL-222` and `Set-Cookie: lastPage=implicit`.

Program

■ Lab record: every tab is one file of the answer. Write all of them.

`implicit.jsp`

implicit.jsp

```
1 <%-- Q11: src/main/webapp/implicit.jsp
2     out, request, response, session, pageContext; exception is shown by err
3     Open /implicit.jsp?fail=1 to trigger the exception path. --%>
4 <%@ page contentType="text/html; charset=UTF-8" language="java" errorPage="er
5 <%
6     // response: set a header and a cookie before any output is flushed
7     response.setHeader("X-Lab", "MCSL-222");
8     response.addCookie(new jakarta.servlet.http.Cookie("lastPage", "implicit
9
10    // session: count visits for this browser
11    Integer visits = (Integer) session.getAttribute("visits");
12    visits = visits == null ? 1 : visits + 1;
13    session.setAttribute("visits", visits);
14
15    // pageContext: attribute in page scope, and a shortcut to the other sco
16    pageContext.setAttribute("pageNote", "only visible on this page");
17    pageContext.setAttribute("appNote", "visible to every page", jakarta.ser
18 %>
19 <!DOCTYPE html>
20 <html>
21 <head><meta charset="UTF-8"><title>Implicit Objects</title></head>
22 <body>
23     <h2>JSP Implicit Objects</h2>
24
25     <h3>1. out</h3>
26     <% out.println("<p>Written with out.println(). Buffer size: " + out.getB
27         + " bytes, remaining: " + out.getRemaining() + "</p>"); %
28
29     <h3>2. request</h3>
30     <p>Method: <%= request.getMethod() %>, URI: <%= request.getRequestURI() %
31         client IP: <%= request.getRemoteAddr() %>, name parameter: <%= reques
32
33     <h3>3. response</h3>
34     <p>Content type set to <%= response.getContentType() %>; header X-Lab an
35
36     <h3>4. session</h3>
37     <p>Session id: <%= session.getId() %>, visits: <%= visits %>, new: <%= s
38
39     <h3>5. pageContext</h3>
40     <p>page scope: <%= pageContext.getAttribute("pageNote") %></p>
41     <p>application scope via pageContext: <%= pageContext.findAttribute("app
42     <p>session via pageContext: <%= pageContext.getSession().getId().equals(
43
```

```
44     <h3>6. exception</h3>
45     <p><a href="implicit.jsp?fail=1">Click to divide by zero</a>; error.jsp
46     <%
47         if ("1".equals(request.getParameter("fail")))) {
48             int zero = 0;
49             out.println(10 / zero);    // ArithmeticException goes to errorPa
50         }
51     %>
52 </body>
53 </html>
```

Output

Checked by reading. Second visit:

```
1 JSP Implicit Objects
2 1. out
3 Written with out.println(). Buffer size: 8192 bytes, remaining: 7810
4 2. request
5 Method: GET, URI: /JspLab/implicit.jsp, client IP: 0:0:0:0:0:0:0:1, name par
6 3. response
7 Content type set to text/html;charset=UTF-8; header X-Lab and cookie lastPag
8 4. session
9 Session id: 7D2C9F1A4B8E3C6D0A5F2E9B1C4D7A8E, visits: 2, new: false
10 5. pageContext
11 page scope: only visible on this page
12 application scope via pageContext: visible to every page
13 session via pageContext: true (same object as session)
14 6. exception
15 Click to divide by zero; error.jsp shows the exception object.
```

After clicking the link (`implicit.jsp?fail=1`) the browser shows `error.jsp` : `Exception type: java.lang.ArithmeticException` , `Message: / by zero` , `Thrown from: org.apache.jsp.implicit_jsp._jspService(implicit_jsp.java:142)` .

Explanation

Object	Type	Example use in the page
<code>out</code>	<code>JspWriter</code>	<code>out.println</code> , buffer size and remaining space
<code>request</code>	<code>HttpServletRequest</code>	method, URI, remote address, <code>name</code> parameter
<code>response</code>	<code>HttpServletResponse</code>	<code>setHeader</code> , <code>addCookie</code> , content type
<code>session</code>	<code>HttpSession</code>	id, visit counter attribute, <code>isNew()</code>
<code>pageContext</code>	<code>PageContext</code>	attributes in page and application scope, <code>findAttribute</code> , <code>getSession()</code>
<code>exception</code>	<code>Throwable</code>	class name, message, first stack frame in <code>error.jsp</code>

- The six objects are local variables that Tomcat declares at the top of `_jspService` ; that is why they exist without any import.
- `exception` exists only in a page marked `isErrorPage="true"` . The page that can fail names its handler with `errorPage="error.jsp"` ; on an uncaught exception Tomcat forwards to it and sets the `exception` object. `error.jsp` also serves the `web.xml` error pages of Question 12, where it reads the status code from the `jakarta.servlet.error.status_code` attribute.
- `pageContext.findAttribute` searches page, request, session then application scope in that order; it is what EL uses when it resolves a bare name.

Question 12

Problem Statement

■ Write in lab record

Create a JSP Project implementing all the above (Session 1 and Session 2) concepts. Login Form, CRUD operation of Student details, Session Management with exception handling using Servlet and JSP. Make necessary assumptions required.

Solution

■ Write in lab record

Assumptions

- Single administrator account `admin` / `ignou123` hard-coded in `LoginServlet`; a Users table with hashed passwords is left for Session 9 (Spring Security). Database, `Student` and `StudentDao` are those of Session 1, Question 5.
- Session timeout is 10 minutes; after that any request to `/students` returns to the login page with a message.
- All database and conversion errors surface either as a red message on the list page (expected errors such as a duplicate enrolment number) or on `error.jsp` (anything unexpected, through `web.xml`).

Steps

1. In `JspLab` copy `Student.java` and `StudentDao.java` from Session 1 into package `ignou`.
2. Add `LoginServlet`, `LogoutServlet`, `StudentController` to package `ignou`.
3. Add `login.jsp` in `src/main/webapp/` (Question 11's `error.jsp` is reused as is). Create folder `src/main/webapp/WEB-INF/views/` and add `students.jsp` and `student-form.jsp` there.
4. Replace `WEB-INF/web.xml` with the listing below.
5. Run. `http://localhost:8080/JspLab/` opens `login.jsp`. Try a wrong password, then the correct one. Add, edit and delete a student as in Session 1.
6. Click Logout, then type `http://localhost:8080/JspLab/students` directly: you land on the login page with "Please login first". Open `/JspLab/nothing.jsp` for the 404 branch of `error.jsp`; stop MySQL and open the list for the exception branch.

Program

- Lab record: every tab is one file of the answer. Write all of them.

web.xml

web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- src/main/webapp/WEB-INF/web.xml for the JspLab project (Session 2).
3      Servlet URLs come from @WebServlet annotations; this file adds the
4      welcome page, the session timeout and the error pages used by Q12. -->
5 <web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
6          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7          xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jak
8          version="6.0">
9
10    <display-name>JspLab</display-name>
11
12    <welcome-file-list>
13      <welcome-file>login.jsp</welcome-file>
14    </welcome-file-list>
15
16    <session-config>
17      <session-timeout>10</session-timeout> <!-- minutes of inactivity -->
18      <cookie-config>
19        <http-only>true</http-only>
20      </cookie-config>
21    </session-config>
22
23    <!-- Any exception that escapes a servlet or JSP lands on error.jsp -->
24    <error-page>
25      <exception-type>java.lang.Throwable</exception-type>
26      <location>/error.jsp</location>
27    </error-page>
28    <error-page>
29      <error-code>404</error-code>
30      <location>/error.jsp</location>
31    </error-page>
32  </web-app>
```

`Student.java` and `StudentDao.java` are the Session 1, Question 5 listings, copied without change.

Output

Checked by reading, not executed. A wrong password returns to `login.jsp` with `Invalid username or password` in red above the form. After a correct login the browser is at

`/JspLab/students` and shows `Logged in as admin | Logout`, the heading `IGNOU Students (3)`, the `Add new student` link and the same three-row table as Session 1, Question 5, each row ending in Edit and a Delete button.

After Save on the edit form the same list reappears with `Student updated` in green. Visiting `/students` after Logout shows the login page with `Please login first`. With MySQL stopped, `error.jsp` shows `Exception type: jakarta.servlet.ServletException`, `Message: Database error: Communications link failure`. A wrong URL shows `HTTP status 404 for /JspLab/nothing.jsp`.

Explanation

- Flow: `login.jsp` (view) posts to `LoginServlet` (controller), which stores `user` in the `HttpSession` and redirects to `StudentController`. The controller loads data through `StudentDao` (model), puts it in request attributes and forwards to a JSP under `WEB-INF/views/`. That is Model-View-Controller with plain servlets and JSP, the shape Spring MVC automates in Sessions 3 to 5.
- Session management: `loggedIn()` runs before every action and checks `session.getAttribute("user")`. Login invalidates any old session first so an attacker cannot plant a known session id (fixation). `web.xml` sets a 10 minute timeout and `http-only` on the cookie; `LogoutServlet` invalidates and redirects.
- Views under `WEB-INF` cannot be requested by URL, so nobody can open `students.jsp` without going through the controller and its login check.
- Exception handling has two levels. Expected failures (duplicate key, bad number) are caught in `doPost` and shown as a message. Anything else is wrapped in `ServletException` and left to the container, which the `error-page` entries route to `error.jsp`; the same page handles 404.
- The views use JSTL and EL only (`c:forEach`, `c:out`, `c:url`, `c:if`, `fn:contains`), no scriptlets; every POST ends in a redirect to `/students` (Post-Redirect-Get).

Viva Questions

✗ Do not copy. Read for understanding and the viva

- **Q:** How is a JSP different from a servlet? **A:** A JSP is translated into a servlet by the container; it is HTML with embedded Java rather than Java with embedded HTML, so it suits the view layer.
- **Q:** What is the JSP life cycle? **A:** Translation to a `.java` file, compilation, class loading, `jspInit()`, `_jspService()` per request, `jspDestroy()`.

- **Q:** Difference between the include directive and `jsp:include` ? **A:** The directive merges source at translation time; the action calls the resource at request time and can pass parameters.
- **Q:** Why prefer JSTL and EL over scriptlets? **A:** Views stay readable, output is escaped by default, no Java in HTML, and designers can edit the page.
- **Q:** How does `errorPage` differ from the `error-page` in `web.xml` ? **A:** `errorPage` is per JSP; `web.xml` mappings apply to the whole application and also cover servlets and HTTP status codes.
- **Q:** Why put JSPs under `WEB-INF` ? **A:** The container never serves `WEB-INF` directly, so the pages can only be reached by a forward from a servlet that has done its checks.
- **Q:** What is the JSTL core URI for Tomcat 10.1? **A:** `jakarta.tags.core` (JSTL 3.0); the old `java.sun.com` URI is for the `javax` versions.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Using the JSTL 1.2 URI or jar on Tomcat 10.1: "The absolute uri cannot be resolved" at translation time.
- Setting a header or calling `c:redirect` after HTML has been written: `IllegalStateException: response already committed`.
- A bean without a public no-arg constructor or with a getter that does not match the property name: `jsp:useBean` or `getProperty` fails at run time.
- Forgetting `isErrorPage="true"`: the `exception` object is undefined and the error page itself fails to compile.
- Leaving `students.jsp` outside `WEB-INF`, which lets anyone skip the login check by typing its URL.

Session Summary

■ Write in lab record

- Project `JspLab` on Tomcat 10.1 with JSTL 3.0 (`jakarta.tags.core`) and `mysql-connector-j`
- Question 6: `datetime.jsp` with date, time, timestamp and 5 second auto-refresh (Refresh header and meta tag)
- Question 7: `scripting.jsp` with declaration, scriptlet and expression elements

- Question 8: `jstl-demo.jsp` using `c:out`, `c:if`, `c:forEach`, `c:choose` / `c:when` / `c:otherwise`, `c:url`, `c:redirect`
- Question 9: `student-list.jsp` reading the IGNOU Student table through JDBC
- Question 10: `StudentBean`, `header.jsp`, `action-demo.jsp`, `bean-view.jsp` using `jsp:include`, `jsp:forward`, `jsp:useBean`, `jsp:setProperty`, `jsp:getProperty`
- Question 11: `implicit.jsp` and `error.jsp` covering `out`, `request`, `response`, `session`, `pageContext`, `exception`
- Question 12: login (`login.jsp`, `LoginServlet`, `LogoutServlet`), `StudentController` with session check, views `students.jsp` and `student-form.jsp` under `WEB-INF/views`, `web.xml` error pages

[✎ Edit this page](#)

[< Previous](#)
Session 1

Session 3 [Next >](#)