



Week 5

JSP: JavaBeans & Java Standard Tag Library (JSTL)

Web Programming 2

Name: Luqman Hakim Bin Aziz Matric #: S66292 Semester: II 2023/2024 Lab: MP1 Demonstrator: Sir Arizal



Lecturers

PUSAT PENGAJIAN INFORMATIK DAN MATEMATIK GUNAAN
(PPIMG), UNIVERSITI MALAYSIA TERENGGANU (UMT)

Revision History

Revision Date	Previous Revision Date	Summary of Changes	Changes Marked
		First Issue	Mohamad Nor Hassan
		Second Issue	Dr Rabiei Mamat Dr Faizah Aplop Dr Fouad Ts Dr Rosmayati Mohemad Fakhrul Adli Mohd Zaki
21/02/2019		Addition of Revision History, Table of Contents, Formatting Cover Page	Fakhrul Adli Mohd Zaki

Table of Contents

Task 1: Using Scriptlet to Access a Simple JavaBeans.....	4
Task 2: Problem Solving using JavaBeans	3
Task 3: Installing JSTL Taglibs	5
Task 4: Using Java Standard Tag Library (JSTL).....	8
Task 5: Using JSP Standard Tag Library	14

Arahan:

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Pusat Pengajian Informatik dan Matematik Gunaan (PPIMG), Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual. Tandakan (✓) setiap langkah yang telah selesai dibuat dan tulis kesimpulan bagi setiap aktiviti yang telah selesai dijalankan.

Instruction:

This laboratory manual is for use by the students of the School of Informatics and Applied Mathematics (PPIMG), Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual. Tick (✓) each step completed and write the conclusions for each completed activity.

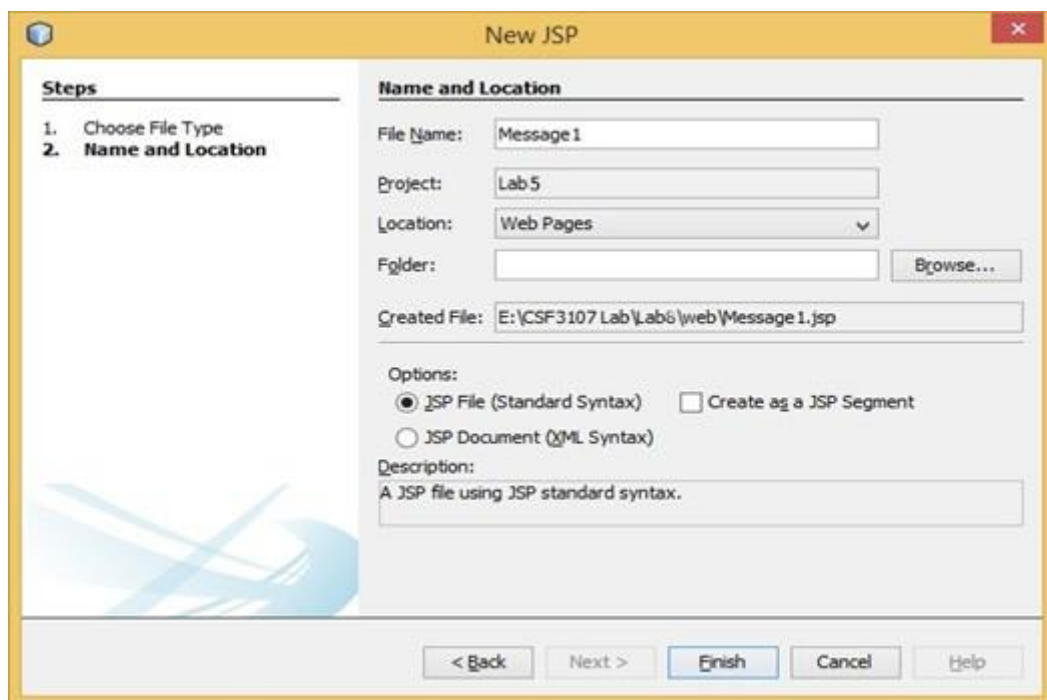
Task 1: Using Scriptlet to Access a Simple JavaBeans

Objective: Use Java Scriptlet to access JavaBeans

Problem Description: Write a JavaBeans that can display message “Welcome to CSF3107 courses....!”.

Estimated time: 20 minutes

1. Go to *Lab5*’s project.
2. Create a new JSP’s file.
3. Type file name as *Message1*.



4. Rename title as *Using JSP Scriptlet*.

5. Rename `<h1>` as *Using JSP Standard Action to call JavaBeans*.

```
9  <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Using JSP Scriptlet </title>
13 </head>
14 <body>
15     <h1>Using JSP Scriptlet to call JavaBeans</h1>
16 </body>
17 </html>
```

6. Use JSP page directive to include the information such as content type, page info, language, lab8.com package and use java util API.

```
1  <!--
2      Document      : message1
3      Created on    : 18-Apr-2016, 16:13:08
4      Author       : Mohamad Nor Hassan
5  -->
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <%@page language="java"%>
9  <%@page info="Using JSP Standard Action to call JavaBeans"%>
10 <%@page import="java.util.Date, lab5.com.Message"%>
```

7. Use a Java scriptlet to create an object for *Message* class.

```
20 <%
21     //Create an object..
22     Message objMsg = new Message();
23
24 %>
```

8. Then, assign the value as *"Welcome to CSF3107 course....!"* via setter method *setMsg(String msg)*.

```
24 //Assign value..
25 objMsg.setMsg("Welcome to CSF3107 course....!");
```

9. Use *getMsg()*'s method to display the message at paragraph.

```
27 //Display value...
28 out.println("<p>" + objMsg.getMsg() + "</p>");
```

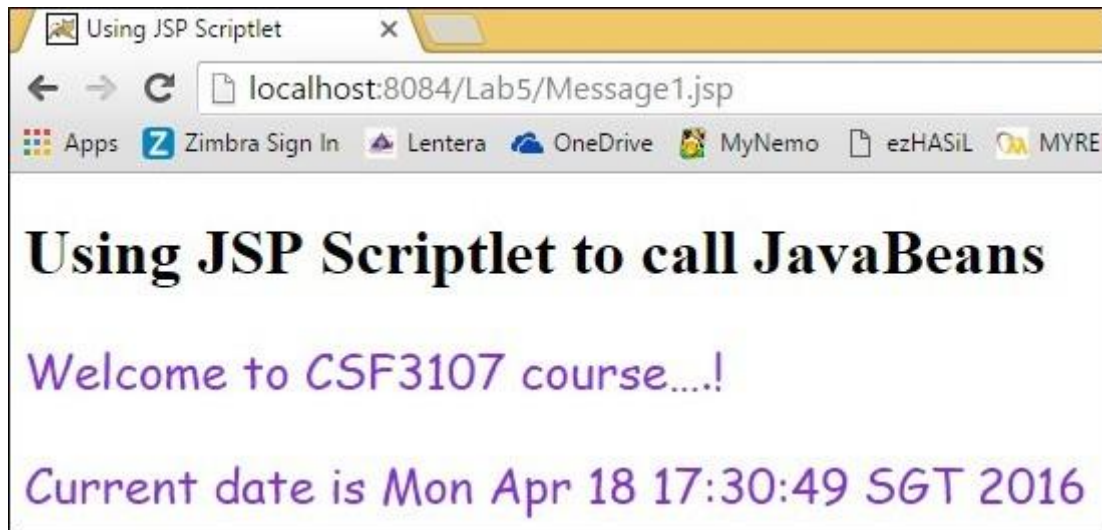
10. Add current date to the next paragraph.

```
30 //Add date..  
31 out.println("<p>Current date is " + new java.util.Date() + "</p>");
```

11. Save *Message1.jsp*

12. Compile and run *Message1.jsp*.

13. You should get the following output.



Reflection

1. What you have learnt from this exercise?

This exercise taught me how to use JSP scriptlets to create and manipulate JavaBeans within JSP pages. It demonstrates how to set and retrieve properties of a JavaBean (Message) with embedded Java code. This method combines Java code and HTML, making the code less maintainable than using JSP Standard Actions.

2. Explain the differences when calling JavaBeans using JSP Standard Action and Java Scriptlet.

JSP Standard Actions use XML-like tags to interact with JavaBeans in a cleaner, more readable manner, promoting separation of concerns and improving maintainability. In contrast, Java Scriptlets combine business logic and presentation by embedding raw Java code within the JSP, resulting in less maintainable and error-prone code. To ensure a clearer and more maintainable codebase, it is recommended to use JSP Standard Actions rather than Scriptlets.

Task 2: Problem Solving using JavaBeans

Objective: Use JavaBeans in JavaScriptlet

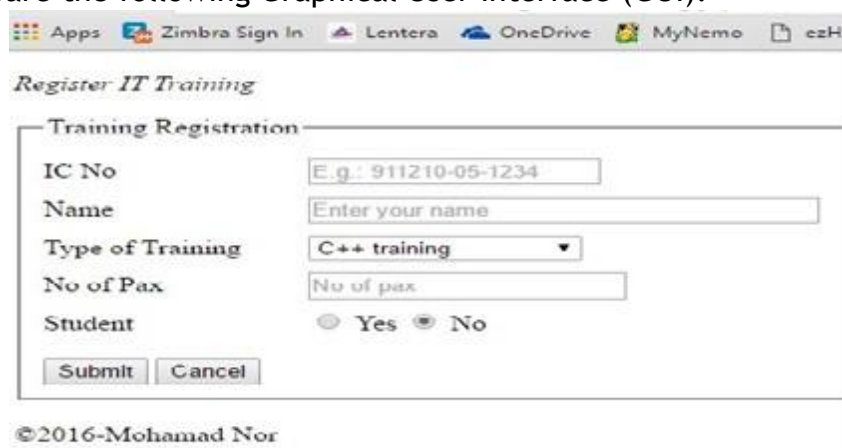
Problem Description: Create sample web form to register IT's training based on the these fees;
C++ training (values as "1")= RM3000 per pax
Java for beginner (values as "2")= RM3000 per pax
HTML5 (values as "3")= RM2800 per pax
Java EEE (values as "4")= RM5500 per pax
Android Programming (values as "5")= RM3200 per pax
Student = Yes (values as "1") and No (Values as "0")
Student will entitle 10% discount
You must cater all front-end validation.

Estimated time: 60 minutes

1. Choose Project *Lab5*.
2. Create a new JSP's file.



3. Type file name as *registerTraining*.
4. Prepare the following Graphical User Interface (GUI).

A screenshot of a web browser displaying a form titled 'Register IT Training'. The form has a header 'Training Registration' and several input fields: 'IC No' with a placeholder 'E.g.: 911210-05-1234', 'Name' with a placeholder 'Enter your name', 'Type of Training' with a dropdown menu showing 'C++ training', 'No of Pax' with a placeholder 'No of pax', and 'Student' with radio buttons for 'Yes' and 'No'. At the bottom are 'Submit' and 'Cancel' buttons. The footer of the form reads '©2016-Mohamad Nor'.

5. Create a *Register* JavaBeans to cater the business requirements and store it into package *lab8.com*.
6. Create a new file name known as *processTraining.jsp*.

7. Attached the *pocessTraining.jsp*'s form to *registerTraining.jsp*.
8. Open *pocessTraining.jsp*'s and use Java Scriptlet to invoke *Register* JavaBeans.
9. Output will appear in web browser.



Coding registerTraining.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="lab5.com.TrainingRegistration"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Register for Training</title>
    <script>
      function validateForm() {
        var icNo = document.forms["registrationForm"]["icNo"].value;
        var name = document.forms["registrationForm"]["name"].value;
        var trainingType = document.forms["registrationForm"]["trainingType"].value;
        var numOfPax = document.forms["registrationForm"]["numOfPax"].value;
        var isStudent = document.forms["registrationForm"]["isStudent"].value;

        if (icNo === "" || name === "" || trainingType === "" || numOfPax === "" || isStudent === "") {
          alert("Please fill in all fields");
          return false;
        }

        if (isNaN(numOfPax)) {
          alert("Number of pax must be a valid number");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <!-- Registration Form -->
  </body>
</html>
```

```

    }

    if (numOfPax < 0) {
        alert("Number of pax cannot be negative");
        return false;
    }

    return true;
}
</script>
</head>
<body>
    <h1>Register for Training</h1>
    <fieldset>
        <legend>Training Registration</legend>
        <form name="registrationForm" action="processTraining.jsp" onsubmit="return validateForm()"
method="post">
            <label for="icNo">IC No:</label>
            <input type="text" id="icNo" name="icNo"><br><br>

            <label for="name">Name:</label>
            <input type="text" id="name" name="name"><br><br>

            <label for="trainingType">Type of Training:</label>
            <select id="trainingType" name="trainingType">
                <option value="">Select Training</option>
                <option value="1">C++ training</option>
                <option value="2">Java for beginner</option>
                <option value="3">HTML5</option>
                <option value="4">Java EE</option>
                <option value="5">Android Programming</option>
            </select><br><br>

            <label for="numOfPax">Number of Pax:</label>
            <input type="text" id="numOfPax" name="numOfPax"><br><br>

            <label>Student:</label>
            <input type="radio" id="isStudentYes" name="isStudent" value="1">
            <label for="isStudentYes">Yes</label>
            <input type="radio" id="isStudentNo" name="isStudent" value="0">
            <label for="isStudentNo">No</label><br><br>

        <%!
        public String getTrainingTypeName(String trainingType) {
            switch (trainingType) {
                case "1":
                    return "C++ training";
                case "2":

```

```

        return "Java for beginner";
    case "3":
        return "HTML5";
    case "4":
        return "Java EEE";
    case "5":
        return "Android Programming";
    default:
        return "Unknown";
    }
}
%>


</form>
</fieldset>
</body>
</html>

```

Coding processTraining.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="lab5.com.TrainingRegistration"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Processing Training Registration</title>
    </head>
    <body>
        <%
            String icNo = request.getParameter("icNo");
            String name = request.getParameter("name");
            String trainingType = request.getParameter("trainingType");
            int numOfPax = Integer.parseInt(request.getParameter("numOfPax"));
            boolean isStudent = request.getParameter("isStudent").equals("1");

            TrainingRegistration registration = new TrainingRegistration();
            registration.setIcNo(icNo);
            registration.setName(name);
            registration.setTrainingType(trainingType);
            registration.setNumOfPax(numOfPax);
            registration.setStudent(isStudent);

            double totalFee = registration.calculateTotalFee();
        %>

        <h1>Training Registration Acknowledgement</h1>
    </body>
</html>

```

```

<p>IC No: <%= registration.getIcNo() %></p>
<p>Name: <%= registration.getName() %></p>
<p>Type of Training: <%= registration.getTrainingType()%></p>
<p>Number of Pax: <%= registration.getNumOfPax() %> <%= "person/s" %></p>
<p>Student: <%= registration.isStudent() ? "Yes" : "No" %></p>
<p>Total Fee: RM <%= totalFee %></p>
</body>
</html>

```

Coding TrainingRegistration.java

```

package lab5.com;

public class TrainingRegistration {
    private String icNo;
    private String name;
    private String trainingType;
    private int numOfPax;
    private boolean isStudent;

    // Getters and setters
    public String getIcNo() {
        return icNo;
    }

    public void setIcNo(String icNo) {
        this.icNo = icNo;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getTrainingType() {
        return trainingType;
    }

    public void setTrainingType(String trainingType) {
        this.trainingType = trainingType;
    }

    public int getNumOfPax() {
        return numOfPax;
    }
}

```

```

public void setNumOfPax(int numOfPax) {
    this.numOfPax = numOfPax;
}

public boolean isStudent() {
    return isStudent;
}

public void setStudent(boolean student) {
    isStudent = student;
}

// Method to calculate total fee
public double calculateTotalFee() {
    double fee = 0;

    switch (trainingType) {
        case "1": // C++ training
            fee = 3000 * numOfPax;
            break;
        case "2": // Java for beginner
            fee = 3000 * numOfPax;
            break;
        case "3": // HTML5
            fee = 2800 * numOfPax;
            break;
        case "4": // Java EEE
            fee = 5500 * numOfPax;
            break;
        case "5": // Android Programming
            fee = 3200 * numOfPax;
            break;
        default:
            break;
    }

    // Apply discount for students
    if (isStudent) {
        fee *= 0.9; // 10% discount
    }

    return fee;
}
}

```

Output



Register for Training

Training Registration

IC No: 010207030629

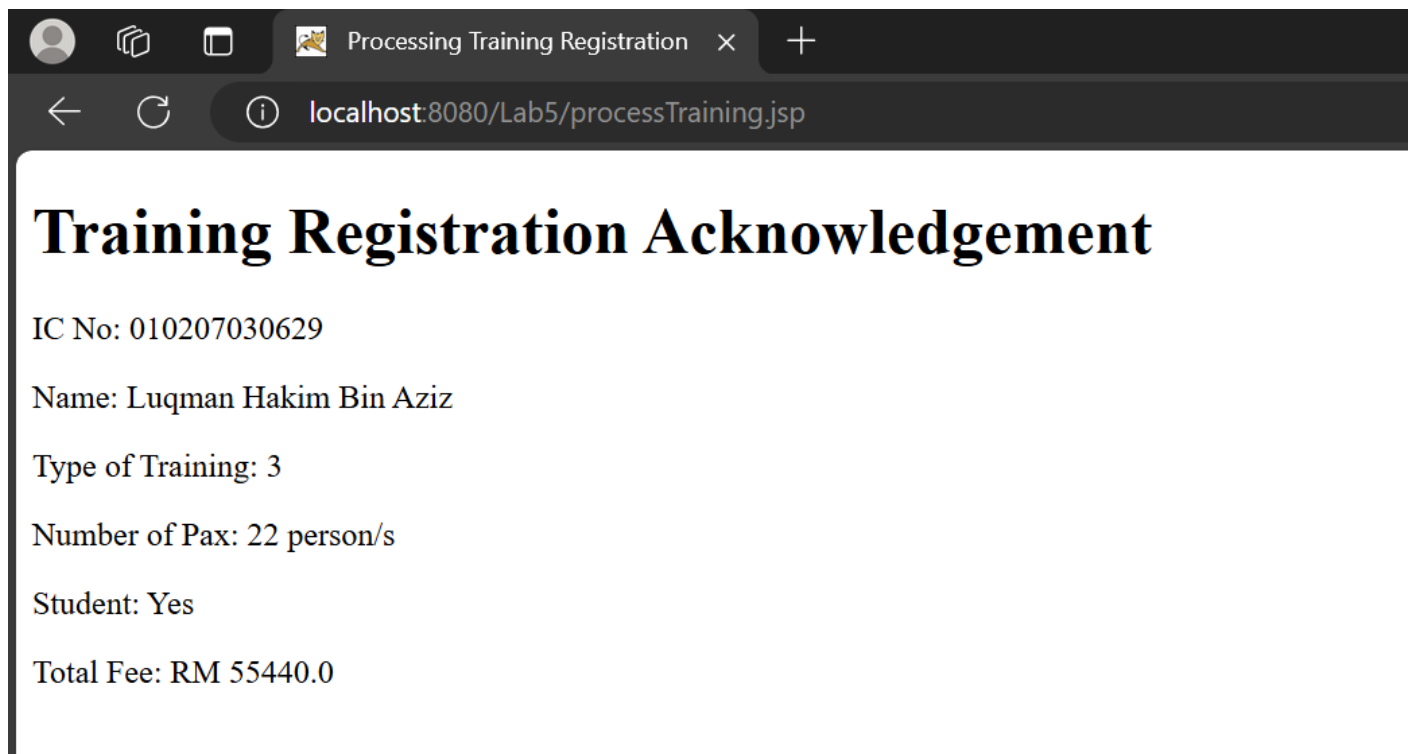
Name: Luqman Hakim Bin Aziz

Type of Training: HTML5

Number of Pax: 22

Student: ☒ Yes ☐ No

Submit



Training Registration Acknowledgement

IC No: 010207030629

Name: Luqman Hakim Bin Aziz

Type of Training: 3

Number of Pax: 22 person/s

Student: Yes

Total Fee: RM 55440.0

Reflection

1. What you have learnt from this exercise?

This exercise taught me how to perform client-side form validation with JavaScript to ensure that user inputs are correct. I also learned how to use JSP to handle form submissions and create JavaBeans to process data. In addition, I learned how to dynamically calculate values based on user input and display the results using JSP.

2. Describe the steps how you construct *Register* JavaBeans?

To create a Register JavaBean, define a class with private fields for each attribute, as well as public getter and setter methods for accessing and modifying these fields. To follow JavaBean conventions, ensure that the class has a no-argument constructor.

Task 3: Installing JSTL Taglibs

Objective: Installation of Apache Taglibs

Problem Description: To install JSTL Library

Estimated time: 25 minutes

1. Go to the browser and type URL

<http://tomcat.apache.org/taglibs/index.html>

Apache Taglibs

This project is an open source repository for JSP(tm) Tag Libraries.

In particular, Apache Taglibs hosts the [Apache Standard Taglib](#), an implementation of the JSTL specification. Various versions are available, and a 1.2 implementation is in the works.

2. Click on *Apache Standard Taglib*.

Standard Taglib

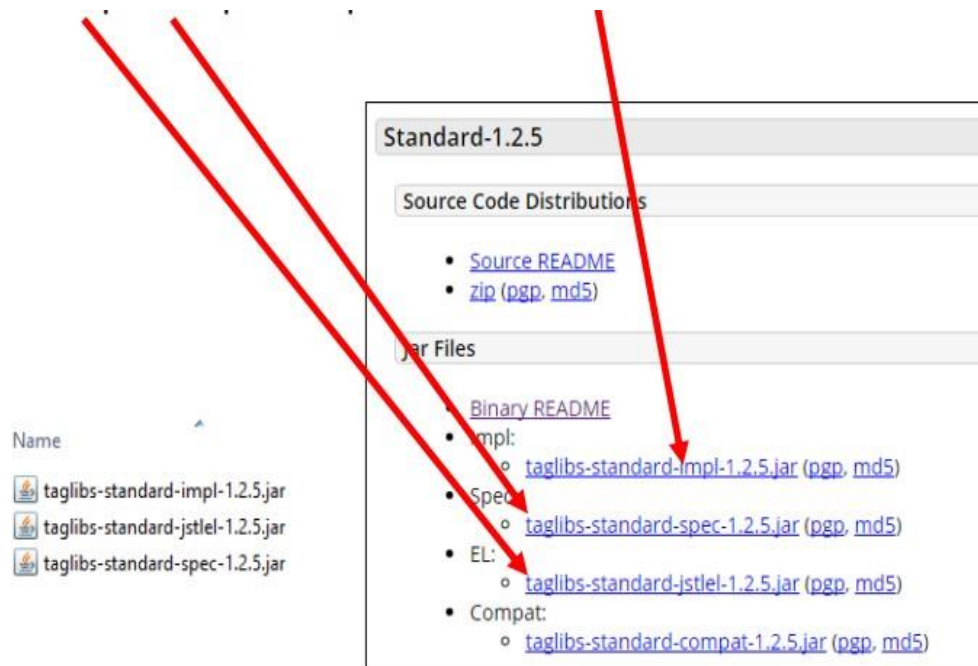
JSP(tm) Standard Tag Library implementations

Apache hosts the Apache Standard Taglib, an implementation of the JSP Standard Tag Library (JSTL) specification. Various versions are available.

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2.3	JSTL 1.2	Servlet 2.5, JavaServer Pages 2.1	download (javadoc)
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	download (javadoc)
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	download (javadoc)

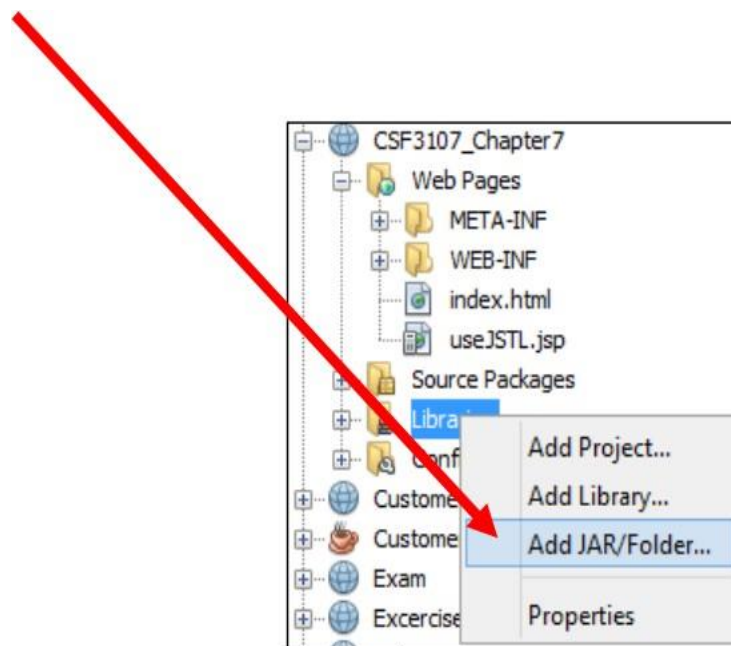
3. Choose Version 1.2 and click download link.
4. Click to jar files for Impl and save the link.

5. Repeat step 3 for Spec and EL.

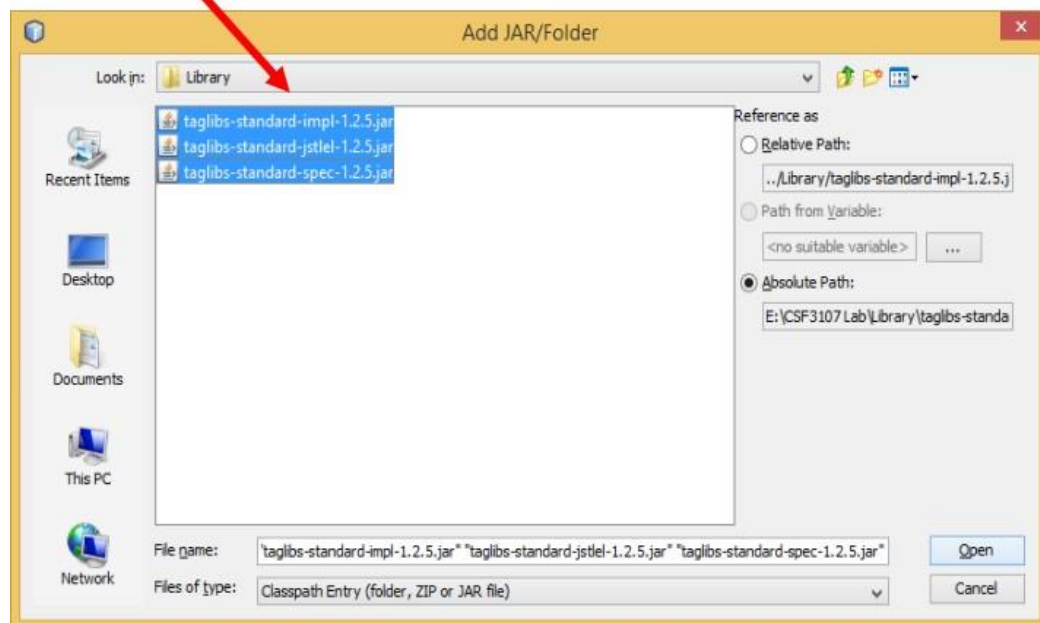


6. Go to your project -> Libraries.

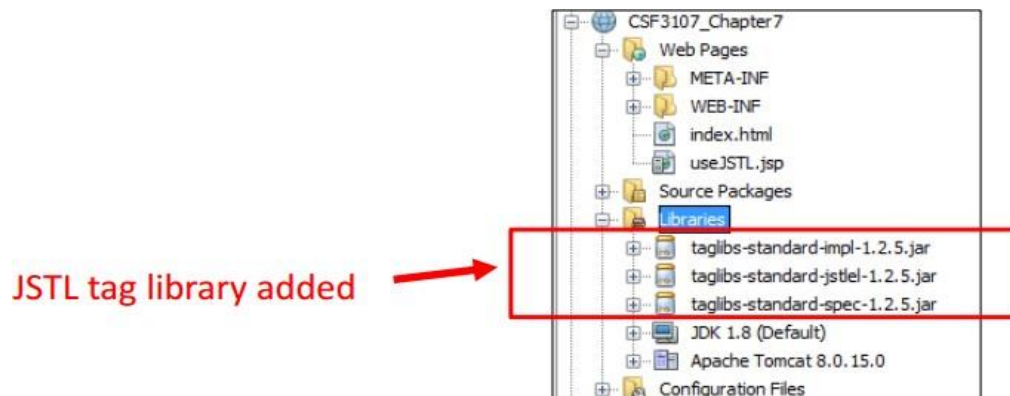
7. Right click your mouse and click to Add JAR/Folder.



8. Choose taglibs-standard-impl-1.2.5.jar, taglibs-standard-jstlel-1.2.5.jar and taglibs-standard-spec-1.2.5.jar and click Open button.



9. You will see all jars files; taglibs-standard-impl-1.2.5.jar, taglibs-standardjstlel-1.2.5.jar and taglibs-standard-spec-1.2.5.jar successfully added in Libraries's folder in NetBeans.



Reflection

What you have learnt from this exercise?

This exercise taught me how to use JSTL to enhance JSP functionality. I also learned how to use JavaBeans to efficiently manage and process form data in a web application.

Task 4: Using Java Standard Tag Library (JSTL)

Objective: Using JSTL core tags directive to retrieve information from one page and display it in another JSP page and format number, currency.

Problem Description:

1. To set the value “Welcome to CSF3107 - Web Programming courses..!” and display the value using JSTL core.
2. To passing information from one page to another using JSTL core
 - i. Create userRegistration.html to key-in information.
 - ii. Create userHandler.jsp to receive information key-in from userRegistration.html.
3. Using JSTL’s fmt to format number and currency.

Estimated time: 30 minutes

Prerequisite: Must attached JSTL’s library to current Project’s directory (Refer to Chapter 7’s notes).

Task 1 Assign value “ Welcome to CS F3107-Web Programming 2 courses” and display the value inside J SP page

1. Go to project *Lab5*.
2. Create a new JSP’s file as *jstlCore1*.

3. Rename the title as *Using JSTL tag library* and `<h1>` as *Use JSTL's features*.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Using JSTL tag library</title>
  </head>
  <body>
    <h1>Use JSTL's features</h1>
  </body>
</html>
```

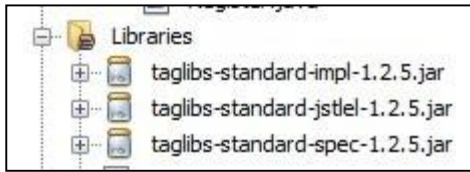
4. Before using JSTL's tag, we need to declare JSTL's library tag using Taglib directive.

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

5. Using JSTL core tag, assign the value *"Welcome to CSF3107 - Web Programming courses..!"* to specific variable and finally, display the message.

```
<body>
  <h1>Use JSTL's features</h1>
  <c:set var="message" value="Welcome to CSF3107 - Web Programming courses..!" />
  <p> <c:out value="${message}" /> </p>
</body>
```

6. Before you compile *jstlCore1.jsp*, please ensure you already attach jar file for JSTL tag library.



7. Compile and run *jstlCore1.jsp* page.

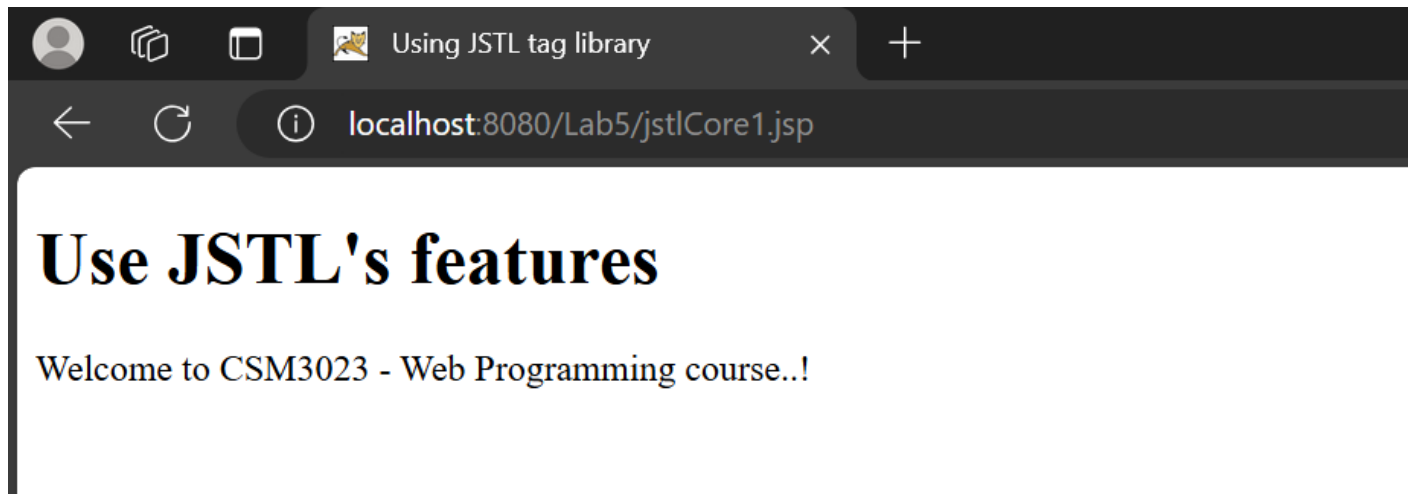
8. You will get the following output.



Coding jstlCore1.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Using JSTL tag library</title>
  </head>
  <body>
    <h1>Use JSTL's features</h1>
    <c:set var="message" value="Welcome to CSM3023 - Web Programming course..!" />
    <p> <c:out value="${message}" /></p>
  </body>
</html>
```

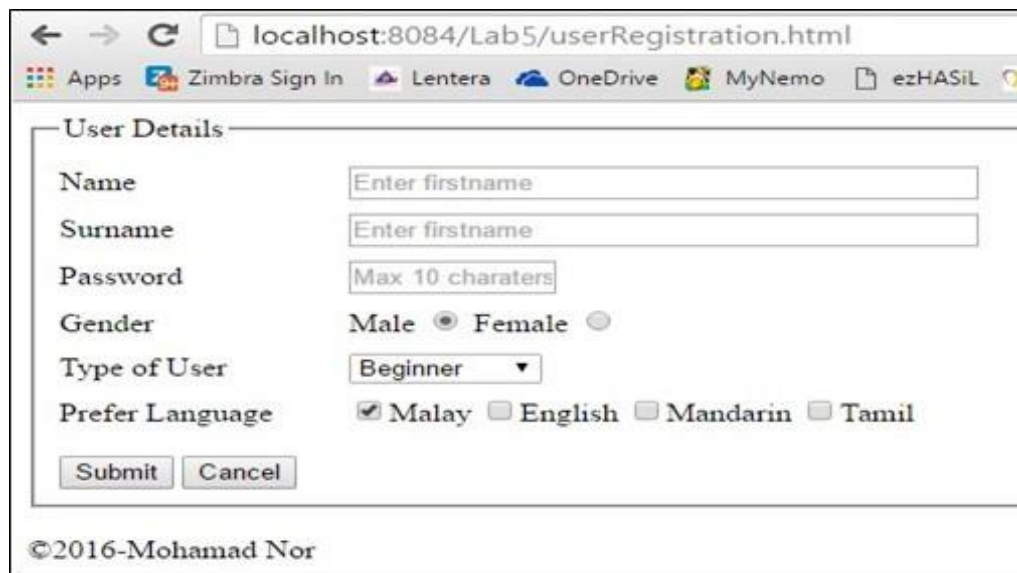
Output



Task 2-Using JSTL's core to request information from one page and display the information

Step 1

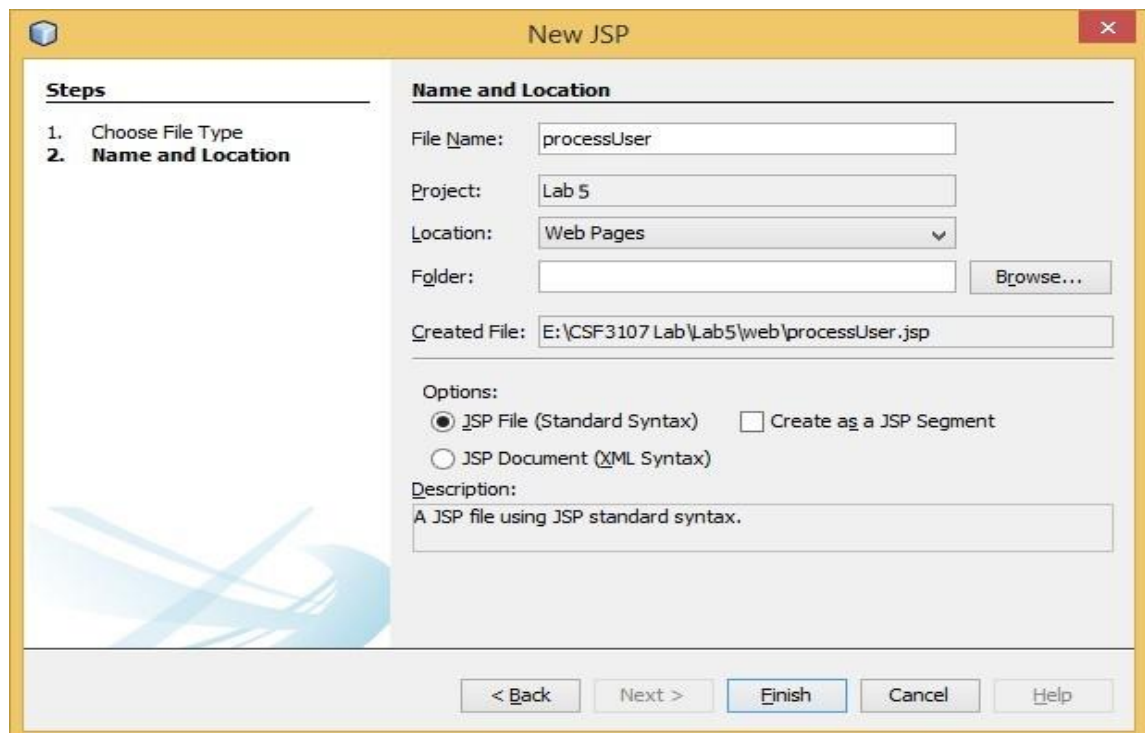
1. Go to Lab8's project.
2. Create HTML's file and rename as userRegistration.
3. Produce the following HTML's form.



Note: Types of user is *Beginner*, *Intermediate* and *Advanced*

Step 2

1. Create JSP's file and rename as *processUser*.



2. Add JSTL taglib directive into the *processUser.jsp*'s page.
3. Retrieve the information by using *c:param* tag and display the information by using *c:out* tag.
4. Compile *userHandler.jsp* page.
5. Run *userRegistration.html* page and key-in the information.

User Details	
Name	Mohamad Nor
Surname	Hassan
Password
Gender	Male <input checked="" type="radio"/> Female <input type="radio"/>
Type of User	Advanced ▼
Prefer Language	<input checked="" type="checkbox"/> Malay <input type="checkbox"/> English <input type="checkbox"/> Mandarin <input type="checkbox"/> Tamil
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>	
©2016-Mohamad Nor	

6. Click *Submit* button.

7. You will get the following output.



Coding userRegistration.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Registration</title>
</head>
<body>
  <fieldset>
    <legend>User Details</legend>
    <form action="processUser.jsp" method="POST">
      <br>
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br><br>
      <label for="surname">Surname:</label>
```

<input type="text" id="surname" name="surname" required>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<label for="gender">Gender:</label>

<input type="radio" id="male" name="gender" value="male" required>

<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="female" required>

<label for="female">Female</label>

<label for="userType">Type of User:</label>

<select id="userType" name="userType">

<option value="beginner">Beginner</option>

<option value="intermediate">Intermediate</option>

<option value="advanced">Advanced</option>

</select>

<label for="preferLanguage">Prefer Language:</label>

<input type="checkbox" id="malay" name="preferLanguage" value="malay">

<label for="malay">Malay</label>

<input type="checkbox" id="english" name="preferLanguage" value="english">

<label for="english">English</label>

<input type="checkbox" id="mandarin" name="preferLanguage" value="mandarin">

<label for="mandarin">Mandarin</label>

<input type="checkbox" id="tamil" name="preferLanguage" value="tamil">

<label for="tamil">Tamil</label>

<input type="submit" value="Submit">

<input type="button" value="Cancel">


```

        </form>

    </fieldset>

    <br>

    <footer>

        &copy Luqman Hakim

    </footer>

</body>

</html>

```

Coding processUser.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Information</title>
</head>
<body>
    <h2>Retrieve info using request parameters & display it using JSP expression</h2>

    <%
        String name = request.getParameter("name");
        String surname = request.getParameter("surname");
        String password = request.getParameter("password");
        String gender = request.getParameter("gender");
        String userType = request.getParameter("userType");
        String preferLanguage = request.getParameter("preferLanguage");
    %>

```

```

<p><b>First Name:</b> <%= name %></p>

<p><b>Surname:</b> <%= surname %></p>

<p><b>Password:</b> <%= password %></p>

<p><b>Gender:</b> <%= gender %></p>

<p><b>Type of user:</b> <%= userType %></p>

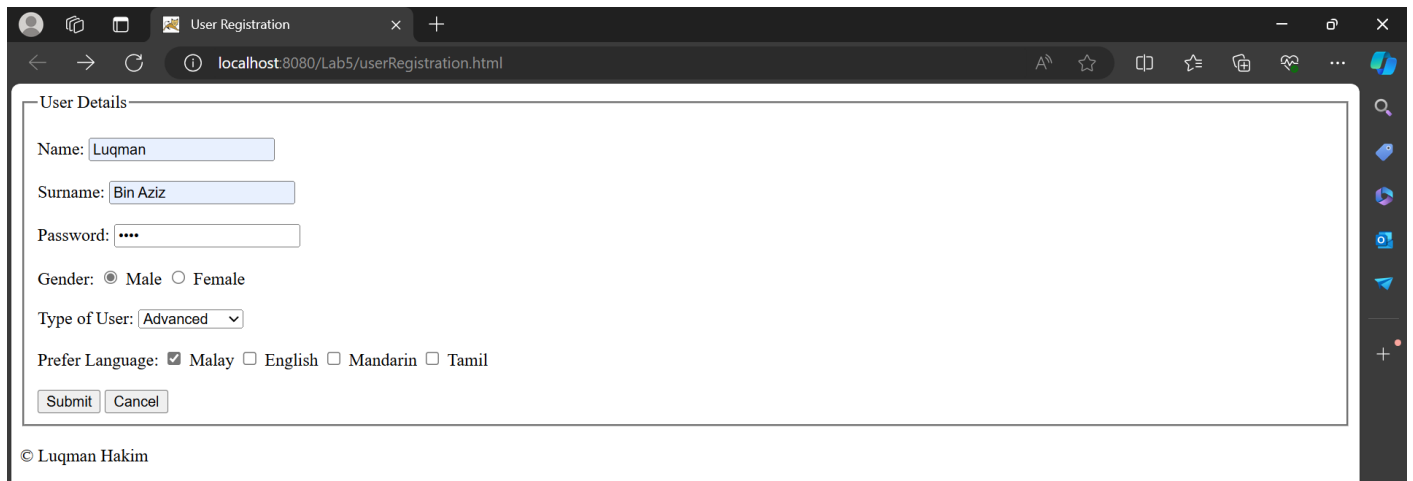
<p><b>Prefer Language:</b> <%= preferLanguage %></p>

</body>

</html>

```

Output



User Registration

localhost:8080/Lab5/userRegistration.html

User Details

Name:

Surname:

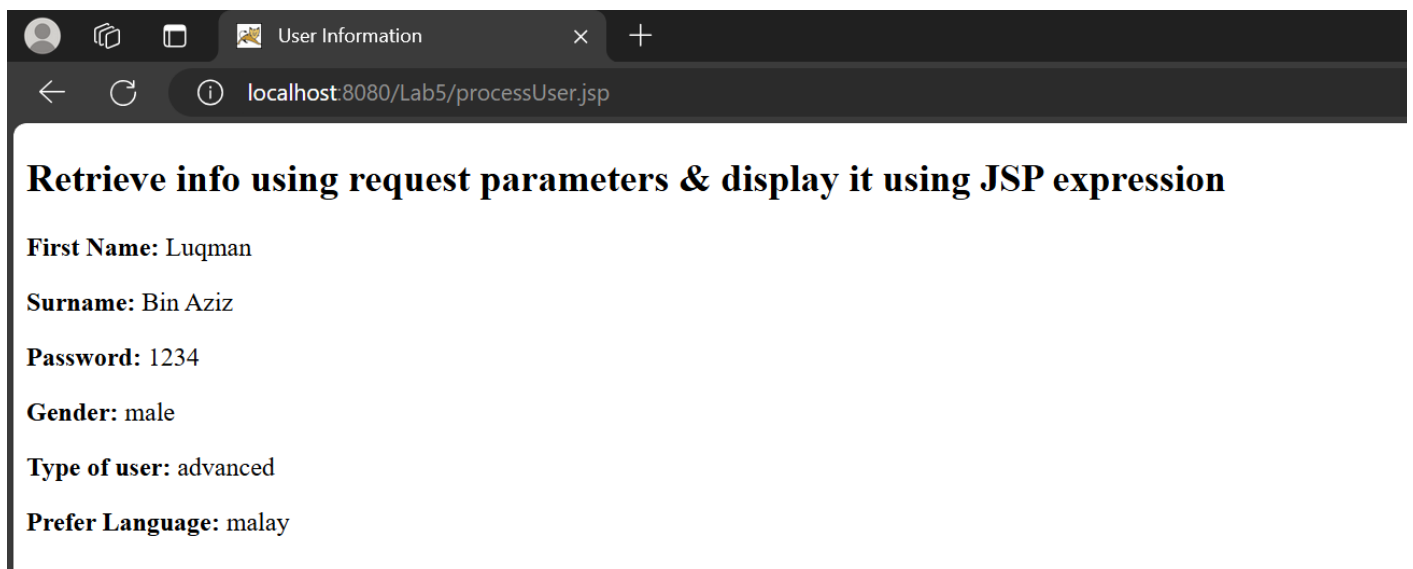
Password:

Gender: ☒ Male ☐ Female

Type of User:

Prefer Language: ☒ Malay ☐ English ☐ Mandarin ☐ Tamil

© Luqman Hakim



User Information

localhost:8080/Lab5/processUser.jsp

Retrieve info using request parameters & display it using JSP expression

First Name: Luqman

Surname: Bin Aziz

Password: 1234

Gender: male

Type of user: advanced

Prefer Language: malay

Task 3 - Using JSTL's fmt to format number and currency

Step 1

1. Create JSP's file and rename as *jstlFormat1*.

New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

< Back Next > **Finish** Cancel Help

2. Add JSTL taglib directive for JSTL *core* and JSTL *formatting* into the current code.
3. Add the following code in your page.

```
<body>
  <h1>Using JSTL formatting tag for formatting</h1>

  <!-- Assign specific number to variable -->
  <c:set var="total" value="2880.4638"/>
  <p>Number to be formatted is <c:out value="${total}" /></p>
  <p>Formatting number as currency with currency code : <fmt:formatNumber type="currency" currencyCode="MYR" value="${total}" /></p>
  <p>Formatting number to the nearest 2 integer digit : <fmt:formatNumber type="number" maxIntegerDigits="2" value="${total}" /></p>
  <p>Formatting number by grouping : <fmt:formatNumber type="number" groupingUsed="true" value="${total}" /></p>
</body>
```

4. Continue to do formatting for
 - 3 decimal places
 - percentage
5. Save *jstlFormat1.jsp*
6. Compile and run *jstlFormat1.jsp*.
7. You will get the following output.



← → ↻ localhost:8084/Lab5/jstlFormat1.jsp

Apps Zimbra Sign In Lentera OneDrive MyNemo ezHASiL MYREN Cloud Pi

Using JSTL formatting tag for formatting

Number to be formatted is 2880.4638

Formatting number as currency with currency code : MYR2,880.46

Formatting number to the nearest 2 integer digit : 80.464

Formatting number by grouping : 2,880.464

Formatting number by 3 decimal places : 2880.464

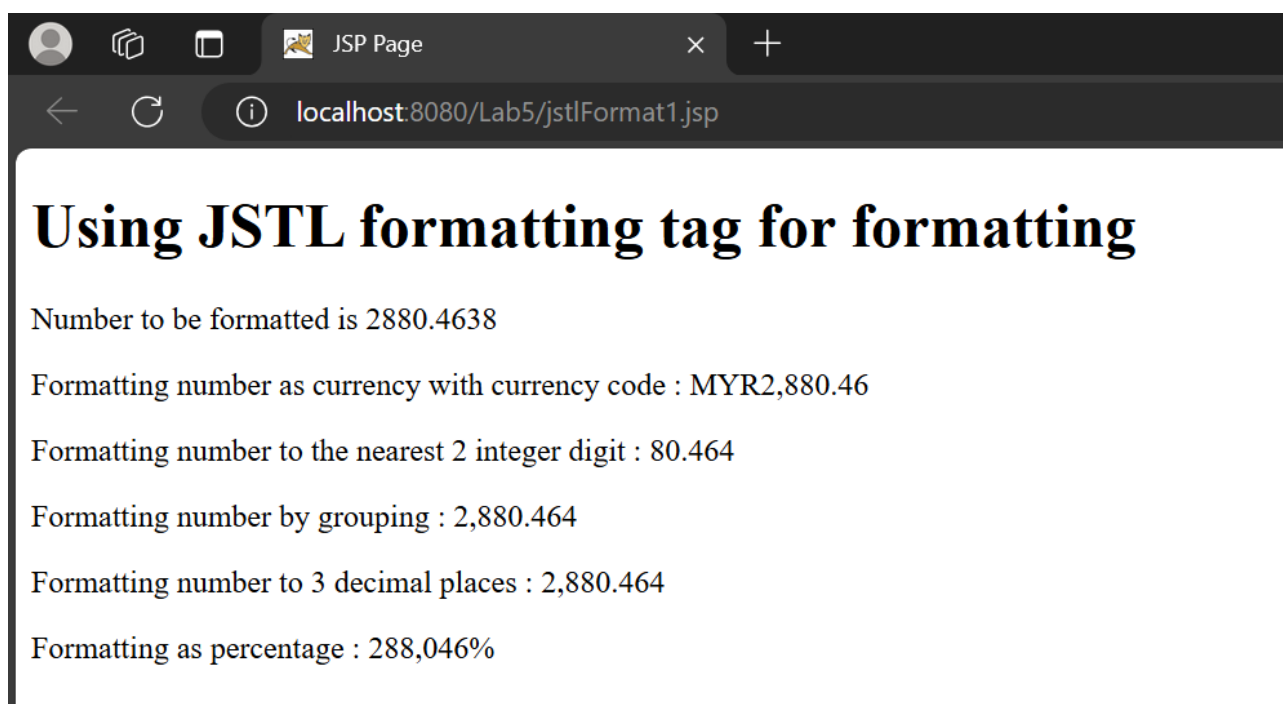
Formatting number by % : 288,046%

Coding jstlFormat1.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Using JSTL formatting tag for formatting</h1>

    <!--Assign specific number to variable -->
    <c:set var="total" value="2880.4638"/>
    <p>Number to be formatted is <c:out value="\${total}"/></p>
    <p>Formatting number as currency with currency code : <fmt:formatNumber type="currency"
currencyCode="MYR" value="\${total}"/></p>
    <p>Formatting number to the nearest 2 integer digit : <fmt:formatNumber type="number"
maxIntegerDigits="2" value="\${total}"/></p>
    <p>Formatting number by grouping : <fmt:formatNumber type="number" groupingUsed="true"
value="\${total}"/></p>
    <p>Formatting number to 3 decimal places : <fmt:formatNumber type="number"
maxFractionDigits="3" value="\${total}"/></p>
    <p>Formatting as percentage : <fmt:formatNumber type="percent" value="\${total}"/></p>
  </body>
</html>
```

Output



Reflection

1. What the purpose of using JSTL's tag library?

Using JSTL's tag library simplifies the creation and maintenance of JSP pages by providing standard tags for common tasks such as iteration, conditionals, and formatting. This reduces the need for Java code in JSPs, making them easier to read and maintain.

2. List **FIVE(5)** categories of JSTL library.

- Core Tags (<c:....>)
- Formatting Tags (<fmt:....>)
- SQL Tags (<sql:....>)
- XML Tags (<x:....>)
- Functions Tags (<fn:....>)

Task 5: Using JSP Standard Tag Library

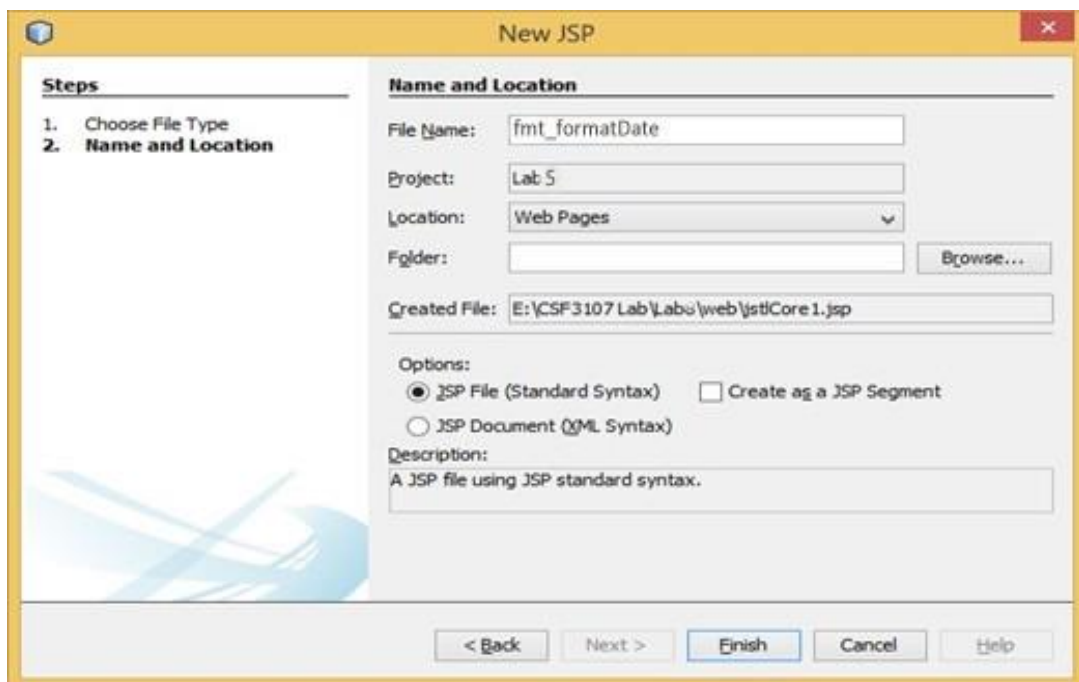
Objective: Using JSTL tags for parsing and formatting the dates.

Problem Description: 1. Using JSTL's `fmt` to format Dates.
2. Using JSTL's `fmt` to parse Dates.

Estimated time: 30 minutes

Task 1 Using JSTL's `fmt` to format Date

1. Go to project *Lab5*.
2. Create a new JSP file as *fmt_formatDate*.



2. Rename the title as *Using JSTL tag library* and `<h2>` as *Use fmt_formatDate features*.

```
<title>fmt:parseDate feature</title>
</head>
<body>
  <h2>fmt:parseDate feature</h2>
  <!-- a Date time string -->
```

4. Before using JSTL's tag, we need to declare JSTL's library tag using Taglib directive.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
```

5. Add JSTL Taglib directive for JSTL core and JSTL formatting into the current code.
6. Add the following code in your page.

```
<c:set var="now" value="<%=new java.util.Date()%>" />
<p>
  Time (fmt:formatDate type="time"):
  <strong>
    <fmt:formatDate type="time" value="{now}" />
  </strong>
</p>
<p>
  Date (fmt:formatDate type="date"):
  <strong>
    <fmt:formatDate type="date" value="{now}" />
  </strong>
</p>
<p>
  Date, Time (fmt:formatDate type="both"):
  <strong>
    <fmt:formatDate type="both" value="{now}" />
  </strong>
</p>
<p>
  Date, Time Short (fmt:formatDate type="both" dateStyle="short"):
  <strong>
    <fmt:formatDate type="both" dateStyle="short" timeStyle="short" value="{now}" />
  </strong>
</p>
```



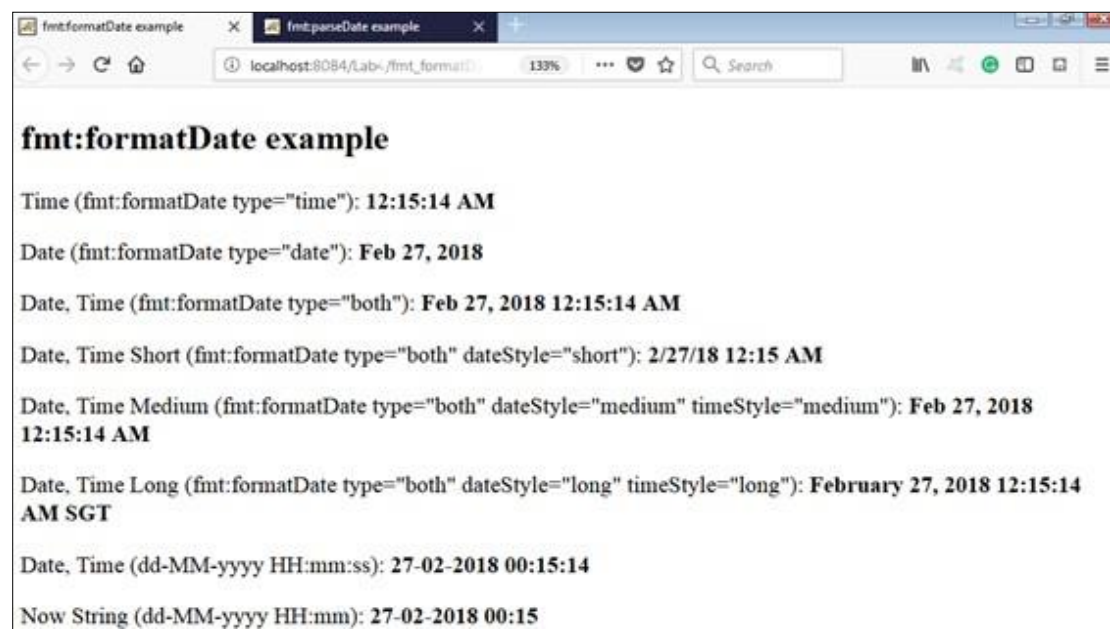
```

<p>
  Date, Time Medium (fmt:formatDate type="both" dateStyle="medium" timeStyle="medium"):
  <strong>
    <fmt:formatDate type="both" dateStyle="medium" timeStyle="medium" value="{now}" />
  </strong>
</p>
<p>
  Date, Time Long (fmt:formatDate type="both" dateStyle="long" timeStyle="long"):
  <strong>
    <fmt:formatDate type="both" dateStyle="long" timeStyle="long" value="{now}" />
  </strong>
</p>
<p>
  Date, Time (dd-MM-yyyy HH:mm:ss):
  <strong>
    <fmt:formatDate pattern="dd-MM-yyyy HH:mm:ss" value="{now}" />
  </strong>
</p>
  <!-- Store in variable -->
<fmt:formatDate pattern="dd-MM-yyyy HH:mm" value="{now}" var="nowString"/>
<p>
  Now String (dd-MM-yyyy HH:mm):
  <strong>
    <c:out value="{nowString}" />
  </strong>
</p>

```

7. Compile and run *fmt_formatDate.jsp* page.

8. You will get the following output.



Coding fmt_formatDate.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>fmt:parseDate feature</title>
  </head>
  <body>
    <h2>fmt:parseDate feature</h2>

    <!-- Declare JSTL's taglib directives -->
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

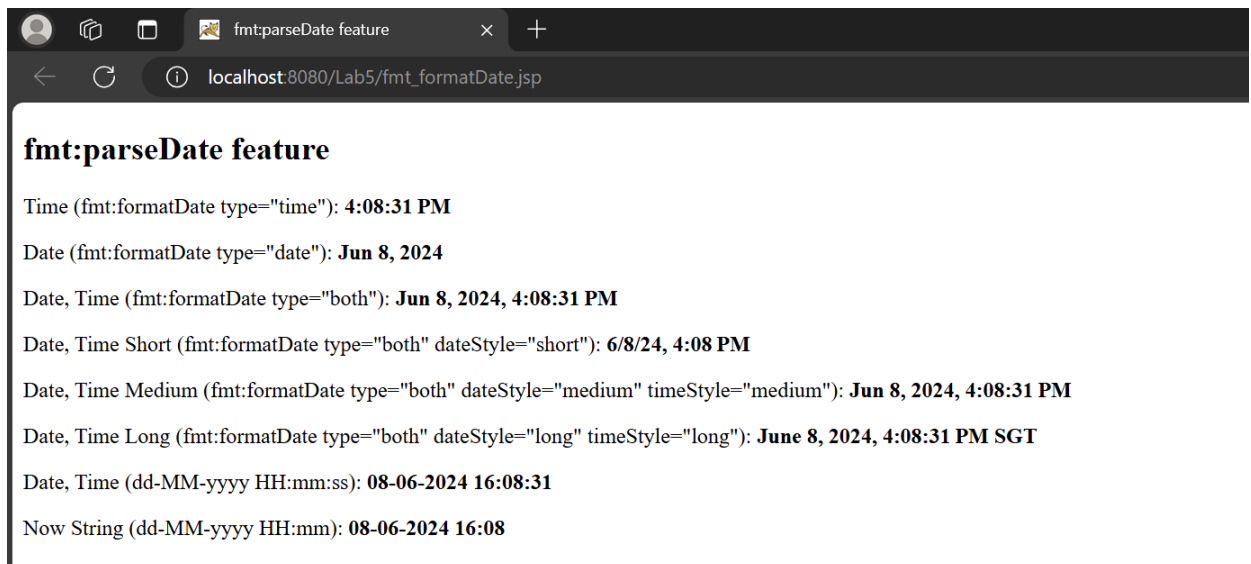
    <c:set var="now" value="<%=new java.util.Date()%>" />
    <p>
      Time (fmt:formatDate type="time"):
      <strong>
        <fmt:formatDate type="time" value="{now}" />
      </strong>
    </p>
    <p>
      Date (fmt:formatDate type="date"):
      <strong>
        <fmt:formatDate type="date" value="{now}" />
      </strong>
    </p>
    <p>
      Date, Time (fmt:formatDate type="both"):
      <strong>
        <fmt:formatDate type="both" value="{now}" />
      </strong>
    </p>
    <p>
      Date, Time Short (fmt:formatDate type="both" dateStyle="short"):
      <strong>
        <fmt:formatDate type="both" dateStyle="short" timeStyle="short" value="{now}" />
      </strong>
    </p>
    <p>
      Date, Time Medium (fmt:formatDate type="both" dateStyle="medium"
timeStyle="medium"):
      <strong>
        <fmt:formatDate type="both" dateStyle="medium" timeStyle="medium" value="{now}"
/>
```

```

        </strong>
    </p>
    <p>
        Date, Time Long (fmt:formatDate type="both" dateStyle="long" timeStyle="long"):
        <strong>
            <fmt:formatDate type="both" dateStyle="long" timeStyle="long" value="{now}" />
        </strong>
    </p>
    <p>
        Date, Time (dd-MM-yyyy HH:mm:ss):
        <strong>
            <fmt:formatDate pattern="dd-MM-yyyy HH:mm:ss" value="{now}" />
        </strong>
    </p>
    <!-- Store in variable -->
    <fmt:formatDate pattern="dd-MM-yyyy HH:mm" value="{now}" var="nowString"/>
    <p>
        Now String (dd-MM-yyyy HH:mm):
        <strong>
            <c:out value="{nowString}" />
        </strong>
    </p>
</body>
</html>

```

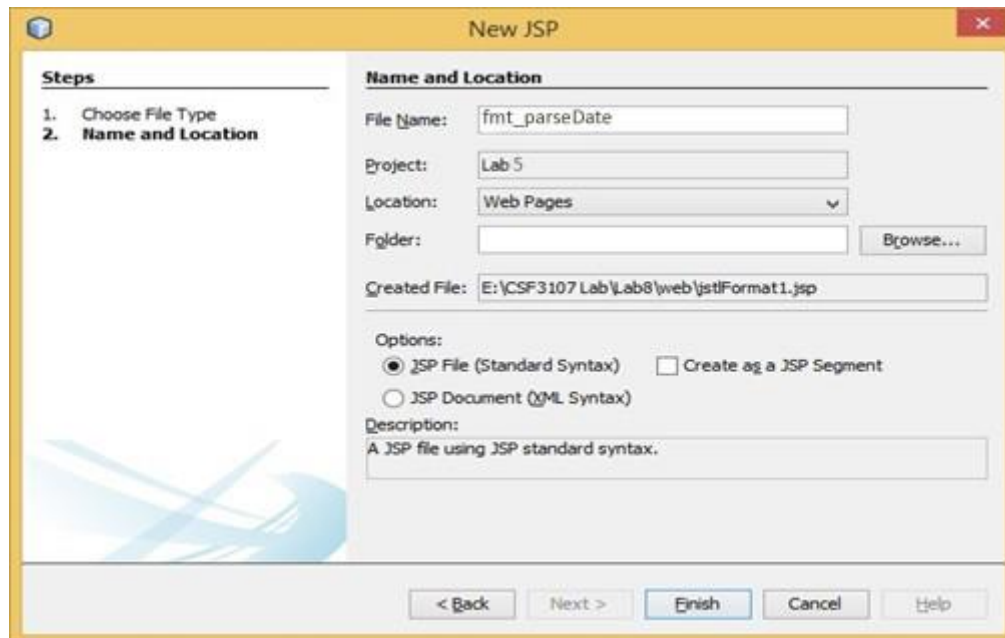
Output



Task 2 - Using JSTL's fmt to Parse Date

Step 1

1. Create JSP's file and rename as *fmt_parseDate*.



2. Add JSTL taglib directive for JSTL *core* and JSTL *formatting* into the current code.
3. Add the following code in your page.

```
<body>
  <c:set var="dateTimeString" value="17-11-2015 11:49" />
  <h4>
    dateTimeString:
    <c:out value="${dateTimeString}" />
  </h4>

  <!-- Parsing a date time string, and store in a variable type of java
  <fmt:parseDate value="${dateTimeString}"
    type="both" var="parsedDatetime" pattern="dd-MM-yyyy HH:mm" />
  <p>
    The date time after parsing:
    <c:out value="${parsedDatetime}" />
  </p>
  <br/>
  <p>
    Date only (dd/MM/yyyy):
    <fmt:formatDate value="${parsedDatetime}" pattern="dd/MM/yyyy" />
  </p>
</body>
```

5. Save *fmt_parseDate.jsp*
6. Compile and run *fmt_parseDate.jsp*.

7. You will get the following output.



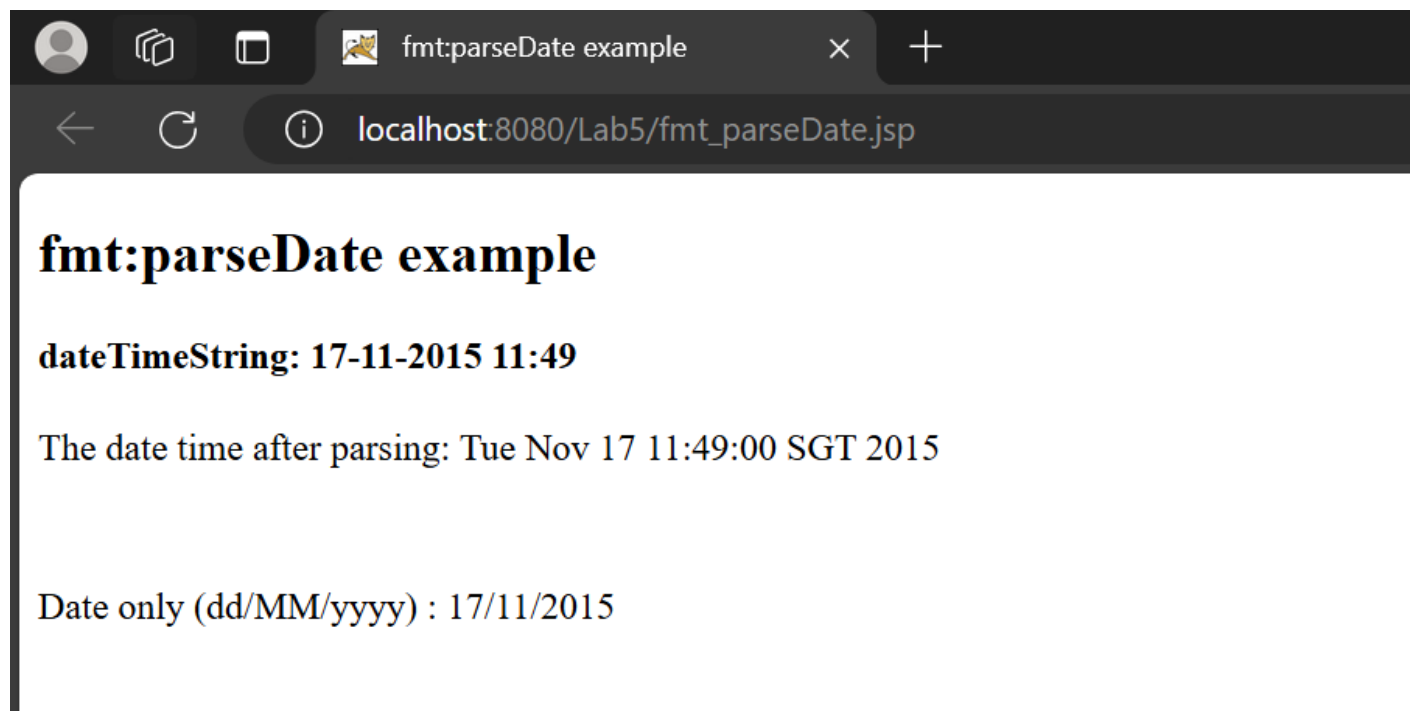
Coding fmt_parseData.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>fmt:parseDate example</title>
  </head>
  <body>
    <h2>fmt:parseDate example</h2>
    <c:set var="dateTimeString" value="17-11-2015 11:49" />
    <h4>
      dateTimeString:
      <c:out value="${dateTimeString}" />
    </h4>

    <!-- Parsing a date time String, and store in a variable type of java-->
    <fmt:parseDate value="${dateTimeString}"
      type="both" var="parsedDatetime" pattern="dd-MM-yyyy HH:mm" />
    <p>
      The date time after parsing:
      <c:out value="${parsedDatetime}" />
    </p>
    <br/>
    <p>
      Date only (dd/MM/yyyy) :
      <fmt:formatDate value="${parsedDatetime}" pattern="dd/MM/yyyy" />
    </p>
  </body>
```

</html>

Output



Reflection

1. What you have learnt from this exercise?

This exercise taught me how to use the JSTL tags 'fmt:formatDate' and 'fmt:parseDate' to format and parse dates in JSP. This improves the flexibility and usability of web applications that handle date and time data.

Exercise

1. Write a JSP's form that asks the user to key-in the radius of circle. The program should calculate the area of circle and the perimeter of circle. Finally, use JSTL library to format your result into 3 decimal places.

Coding circle.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Calculate Circle Area and Perimeter</title>
</head>
<body>
  <h2>Calculate Circle Area and Perimeter</h2>
  <form method="post" action="calculate.jsp">
    Enter the radius of the circle: <input type="text" name="radius">
    <input type="submit" value="Calculate">
  </form>
</body>
</html>
```

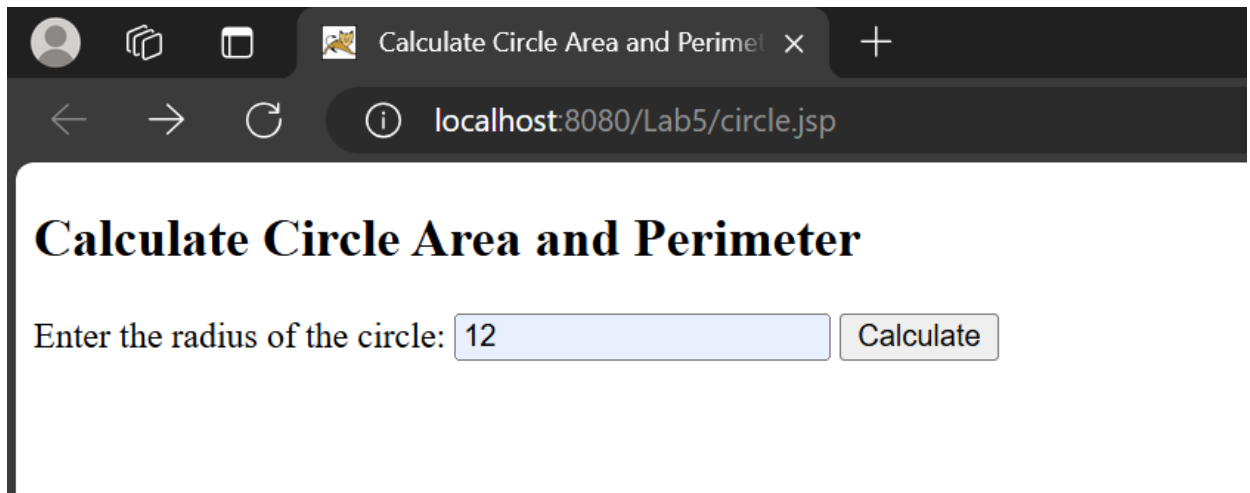
Coding calculate.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Calculate Circle Area and Perimeter</title>
</head>
<body>
  <h2>Results</h2>
  <%-- Retrieve radius from request --%>
  <c:set var="radius" value="${param.radius}" />
  <%-- Calculate area and perimeter --%>
  <c:set var="area" value="${3.14159265 * radius * radius}" />
  <c:set var="perimeter" value="${2 * 3.14159265 * radius}" />

  <p>Area of the circle: <fmt:formatNumber value="${area}" pattern="###.###"/></p>
  <p>Perimeter of the circle: <fmt:formatNumber value="${perimeter}" pattern="###.###"/></p>
```

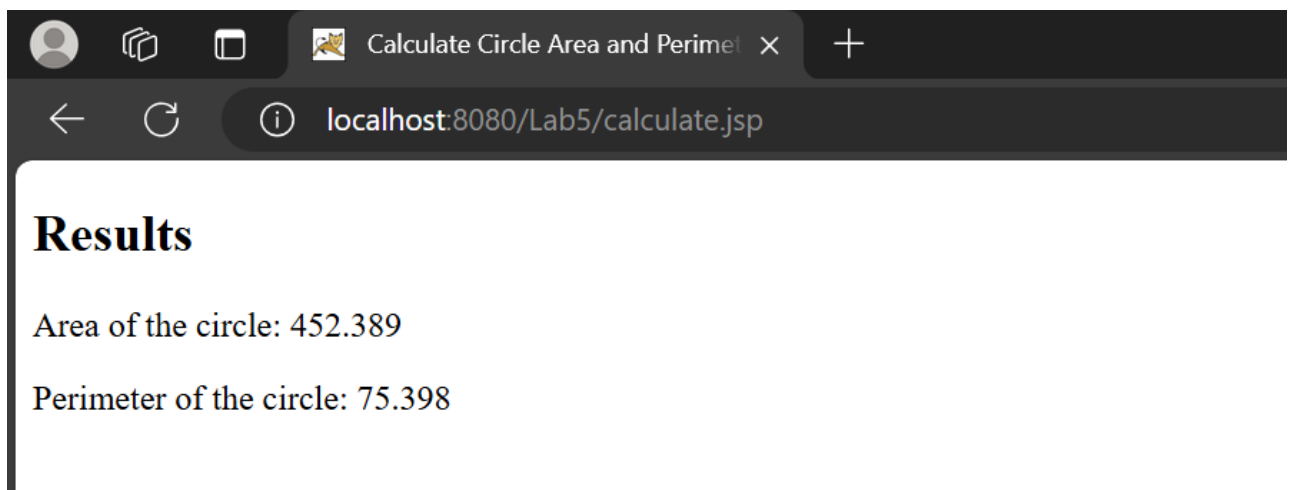
```
</body>
</html>
```

Output



Calculate Circle Area and Perimeter

Enter the radius of the circle:



Results

Area of the circle: 452.389

Perimeter of the circle: 75.398

2. Rahim bought 800 shares of stock at a price of RM10.50 per share. He must pay her stock broker a 5 percent commission for the transaction. Write a web based program that calculates and displays the following:
- The amount paid for the stock alone without the commission.
 - The amount of the commission.
 - The total amount paid (for the stock plus the commission).

You should use JavaBeans to implement business logic and JSTL for display purposes.

Coding brokerage.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@page import="lab5.com.processBrokerage"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>Kimm Trading Broker</title>

    </head>

    <body>

        <h1>Welcome to Kimm Trading Broker!</h1>

        <%

            //initialize all variables needed

            int shares = 800;

            double price = 10.50;

            //instantiate broker object

            processBrokerage broker = new processBrokerage(shares, price);

        %>

        <!-- declare variable using taglibs approach (c:set)-->
```

```
<c:set var="amount" value="<%=broker.getAmountB(shares, price)%>" />
```

```
<c:set var="commission" value="<%=broker.getCommission(shares, price)%>" />
```

```
<c:set var="total" value="<%=broker.getAmountA(shares, price)%>" />
```

```
<!-- output the value using fmt:format to ensure the output is 2 decimal places -->
```

```
<p>Amount (without commission): RM <fmt:formatNumber type="number"
minFractionDigits="2" value="{amount}" /></p>
```

```
<p>Commission charged: RM <fmt:formatNumber type="number" minFractionDigits="2"
value="{commission}" /></p>
```

```
<p>Total amount paid (commission included): RM <fmt:formatNumber type="number"
minFractionDigits="2" value="{total}" /></p>
```

```
</body>
```

```
</html>
```

Coding processBrokerage.java

```
package lab5.com;
```

```
public class processBrokerage {
```

```
    private int shares;
```

```
    private double price;
```

```
    public processBrokerage() {
```

```
    }
```

```
public processBrokerage(int shares, double price) {  
  
    this.shares = shares;  
  
    this.price = price;  
  
}
```

```
public int getShares() {  
  
    return shares;  
  
}
```

```
public void setShares(int shares) {  
  
    this.shares = shares;  
  
}
```

```
public double getPrice() {  
  
    return price;  
  
}
```

```
public void setPrice(double price) {  
  
    this.price = price;  
  
}
```

```
public double getAmountB(int shares, double price){
```

```

        double amountB = shares * price;

        return amountB;
    }

    public double getCommission(int shares, double price){

        double commission;

        commission = shares * price * 0.05;

        return commission;
    }

    public double getAmountA(int shares, double price){

        double amountA = shares * price;

        double commission = shares * price * 0.05;

        return amountA + commission;
    }
}

```

Output

