

Web based Application Development (CSM3023)

Lab Report 5  
JSP: JavaBeans & Java Standard Tag Library (JSTL)

Programme

Computer Science (Software Engeenering)

Prepared by

NurSyaza Amira Binti Selamat

Matric No

S62318

**Prepared for**

Sir Arizal

**Task 1: Using Scriplet to Access a Simple JavaBeans**

Message1.jsp

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

<%@page language="java" %>

<%@page info="Using JSP Standard Action to call JavaBeans" %>

<%@page import="java.util.Date, Lab5.com.Message" %>

<!-- Lab5.com (java package. kene create sendiri) Message (java class dalam package Lab5.com) -->

<!DOCTYPE html>

<html>

<head>

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

<title>Using JSP Scriplet</title>

</head>

<body>

<h1>Using JSP Scriplet to call JavabBeans</h1>

<%

//create an object

Message objMsg = new Message();

//assign value

objMsg.setMsg("Welcome to CSE3023 course!");

//display value

out.println("<p>" + objMsg.getMsg() + "</p>");

//add date

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

%>

</body>

</html>

Message.java

package Lab5.com;

/\*\*

\*

\* @author VivoBook

\*/

public class Message {

private String msg;

public Message() {

// Default constructor

}

public Message(String msg) {

this.msg = msg;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

}



**Reflection:**

1. What you have learnt from this exercise?

* I learnt how to use JSP Scriplet to call JavaBeans

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

|  |  |
| --- | --- |
| **Calling JavaBeans using JSP Standard Action** | **Calling JavaBeans using Java Scriplet** |
| Encapsulates Java logic within specific tags (e.g., <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>). | Embeds Java code directly into JSP files using <% %> tags |
| Enhances code readability by organizing Java code into distinct sections with well-defined tags. | May clutter JSP files with complex Java code, making it harder to understand and maintain. |
| Promotes separation of concerns by keeping Java code separate from HTML markup, improving code readability. | Can lead to mixing presentation and business logic, hindering code maintainability. |
| Encourages the use of JavaBeans, which are reusable and modular components, making code more modular and testable. | May result in code duplication and reduced modularity, as logic is directly embedded in the JSP file. |

**Task 2: Problem Solving using JavaBeans**

registerTraining.jsp

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

<!DOCTYPE html>

<html>

<head>

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

<title>Register Training</title>

</head>

<body>

<h1>Register IT Training</h1>

<form action="processRegister.jsp" method="post">

<fieldset>

<legend>

Training Registration

</legend>

<label for="ICno">IC No</label>

<input type="text" id="icno" name="my\_icno" size="15" placeholder="E.g 01070710139000"><br/><!-- comment -->

<label for="name">Name</label>

<input type="text" id="name" name="my\_name" size="45" placeholder="Enter your name"><br/>

<label for="TypeTrain">Type of Training</label>

<select id="typeTraining" name="type\_training" required>

<option value="1">C++ Training</option>

<option value="2">Java Beginner</option>

<option value="3">HTML5</option>

<option value="4">Java EEE</option>

<option value="5">Android Programming</option>

</select>

<br/>

<label for="pax">No of Pax</label>

<input type="text" id="pax" name="pax" size="15" placeholder="No of Pax"><br/>

<td><label>Student</label></td>

<td><input type="radio" name="cust\_type" value="0" class="radio

button">No</td>

<td><input type="radio" name="cust\_type" value="1" class="radio

button">Yes</td>

<br/>

<input type="submit" id="btnSubmit" value="Submit"/>

<input type="reset" id="btnCancel" value="Cancel"/>

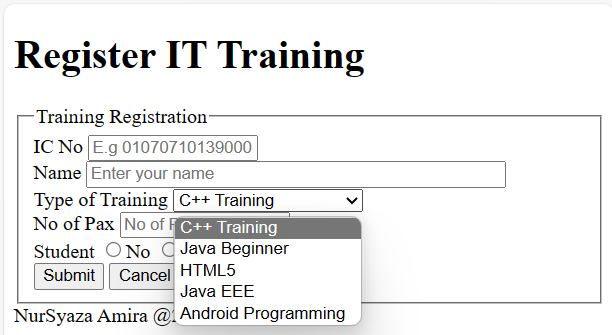
</fieldset>

</form>

</body>

<footer>NurSyaza Amira @2024</footer>

</html>



Register.java

package Lab5.com;

/\*\*

\*

\* @author VivoBook

\*/

public class Register {

private int typeTraining;

private int pax;

private boolean isStudent;

public Register() {

// Default constructor

}

public void setTypeTraining(int typeTraining) {

this.typeTraining = typeTraining;

}

public void setPax(int pax) {

this.pax = pax;

}

public void setIsStudent(boolean isStudent) {

this.isStudent = isStudent;

}

public double calculateTotalCost() {

double cost = 0;

switch (typeTraining) {

case 1:

case 2:

cost = 3000;

break;

case 3:

cost = 2800;

break;

case 4:

cost = 5500;

break;

case 5:

cost = 3200;

break;

}

if (isStudent) {

cost \*= 0.9; // 10% discount for students

}

return cost \* pax;

}

}

processTraining.jsp

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

<%@page language="java" %>

<%@page import="java.util.Map" %>

<%@page import="java.util.HashMap" %>

<%@page import="Lab5.com.Register" %>

<%

// Retrieve form parameters

String icNo = request.getParameter("my\_icno");

String name = request.getParameter("my\_name");

int typeTrainingValue = Integer.parseInt(request.getParameter("type\_training"));

int pax = Integer.parseInt(request.getParameter("pax"));

boolean isStudent = "1".equals(request.getParameter("cust\_type"));

// Map to store training types

Map<Integer, String> trainingTypes = new HashMap<>();

trainingTypes.put(1, "C++ Training");

trainingTypes.put(2, "Java Beginner");

trainingTypes.put(3, "HTML5");

trainingTypes.put(4, "Java EEE");

trainingTypes.put(5, "Android Programming");

// Get the name of the type of training

String typeTraining = trainingTypes.get(typeTrainingValue);

// Create instance of Register JavaBean

Register register = new Register();

register.setTypeTraining(typeTrainingValue);

register.setPax(pax);

register.setIsStudent(isStudent);

// Calculate total cost

double totalCost = register.calculateTotalCost();

%>

<!DOCTYPE html>

<html>

<head>

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

<title>Process Training</title>

</head>

<body>

<h1>Training Registration Summary</h1>

<p>IC No: <%= icNo %></p>

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

<p>Type of Training: <%= typeTraining %></p>

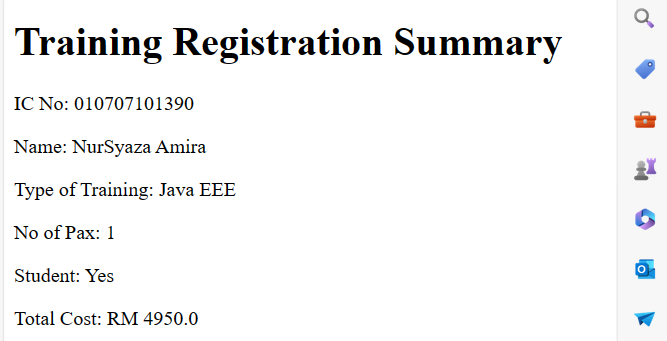
<p>No of Pax: <%= pax %></p>

<p>Student: <%= isStudent ? "Yes" : "No" %></p>

<p>Total Cost: RM <%= totalCost %></p>

</body>

</html>

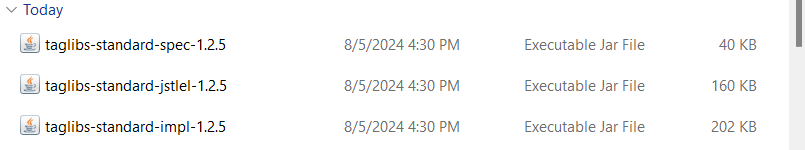


**Reflection:**

1. Describe the steps how you construct Register JavaBeans?

* Define the class
* Define Instance Variables
* Create Constructors
* Create Setter and Getter Method
* Implement Business Logic
* Use Encapsulation

**Task 3: Installing JSTL Taglibs**

****

**Task 4: Using Java Standard Tag Library (JSTL)**

**Task 4.1: Assign value “Welcome to CSE3023-Web Based Application Development courses” and display the value inside JSP Page**

jstlCore1.jsp<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

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

<!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 CSE3023 - Web Based Application Developement"/>

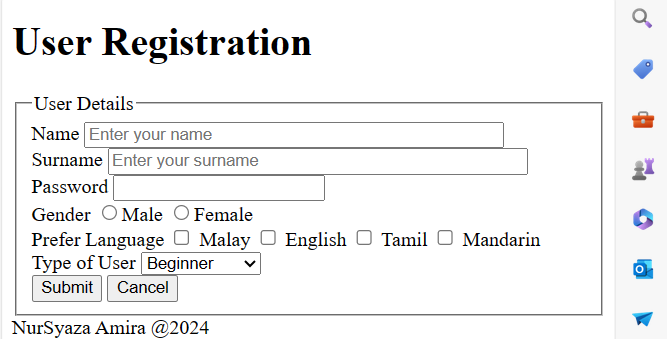
<p><c:out value="${message}"/></p>

</body>

</html>

**Task 4.2: Using JSTL’s core to request information from one page and display the information**

userRegistration.jsp



processUser.jsp

**Task 4.3: Using JSTL’s fmt to format number and currency**

jstlFormat1.jsp

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

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

<%@ 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>JSTL's fmt</title>

</head>

<body>

<h1>Using JSTL formating 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" minFractionDigits="3" maxFractionDigits="3" value="${total}"/></p>

<p>Formatting number as a percentage: <fmt:formatNumber type="percent" value="${total}"/></p>

</body>

<footer>NurSyaza Amira @2024</footer>

</html>

**Reflection:**

1. What the purpose of using JSTL’s tag library?
2. List 5 categories of JSTL library.

**Task 5: Using JSP Standard Tag Library**

**Task 4.1: Using JSTL’s fmt format Date**

fmt\_formatDate.jsp

**Task 4.2: Using JSTL’s fmt to Parse Date**

Fmt\_parseDate.jsp

**Reflection:**

1. What you have learnt from this exercise?

**Exercise 1**

Circle.jsp

processCircle.jsp

**Exercise 2**