A Laboratory Manual for

Advance Java Programming (3160707)

B.E. Semester 7 (Computer)

Directorate of Technical Education,
Gandhinagar, Gujarat

Government Engineering College
Certificate

This is to certify that Mr./Ms. Priyanshi Z Ganvit______________________ Enrollment No. ___230133107022___ of B.E. Semester 6th___ Computer Engineering of this Institute (GTU Code:_____) has satisfactorily completed the Practical / Tutorial work for the subject **Advance Java Programming (3160707)** for the academic year 2025.

Place: __________
Date: __________

Name and Sign of Faculty member

Head of the Department

Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can go through the relevant theory and procedure in advance before the actual performance which creates an interest and students can have basic idea prior to performance. This in turn enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practical.

This manual also provides guidelines to faculty members to facilitate student centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea that how students will be assessed by providing rubrics.

Advance Java Programming is an Elective course. It covers various java based technologies like socket, JDBC, Servlet and JSP which help students to develop enterprise application. Web application based on Java uses Servlet, JSP, JSF etc. To store the data database connectivity and database JDBC component is needed. Networking components are needed to transfer data over network. Model-View-Controller (MVC) architecture gives flexibility and makes the web applications loosely coupled.

Utmost care has been taken while preparing this lab manual however always there is chances of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors if any.

DTE's Vision

- To provide globally competitive technical education
- Remove geographical imbalances and inconsistencies
- Develop student friendly resources with a special focus on girls' education and support to weaker sections
- Develop programs relevant to industry and create a vibrant pool of technical professionals

Institute's Vision

- To create an ecosystem for proliferation of socially responsible and technically sound engineers, innovators and entrepreneurs.

Institute's Mission

- To develop state-of-the-art laboratories and well-equipped academic infrastructure.
- To motivate faculty and staff for qualification up-gradation, and enhancement of subject knowledge.
- To promote research, innovation and real-life problem-solving skills.
- To strengthen linkages with industries, academic and research organizations.
- To reinforce concern for sustainability, natural resource conservation and social responsibility.

Department's Vision

- To create an environment for providing value-based education in Computer Engineering through innovation, team work and ethical practices.

Department's Mission

- To produce computer engineering graduates according to the needs of industry, government, society and scientific community.
- To develop state of the art computing facilities and academic infrastructure.
- To develop partnership with industries, government agencies and R & D organizations for knowledge sharing and overall development of faculties and students.
- To solve industrial, governance and societal issues by applying computing techniques.
- To create environment for research and entrepreneurship.

Programme Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science,engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyzecomplex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineeringproblems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledgeand research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques,resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextualknowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professionalengineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics andresponsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as amember or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activitieswith the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understandingof the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and abilityto engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

- Sound knowledge of fundamentals of computer science and engineering including software and hardware.
- Develop the software using sound software engineering principles having web based/mobile based interface.
- Use various tools and technology supporting modern software frameworks for solving problems having large volume of data in the domain of data science and machine learning.

Program Educational Objectives (PEOs)

- Possess technical competence in solving real life problems related to Computing.
- Acquire good analysis, design, development, implementation and testing skills to formulate simple computing solutions to the business and societal needs.
- Provide requisite skills to pursue entrepreneurship, higher studies, research, and development and imbibe high degree of professionalism in the fields of computing.
- Embrace life-long learning and remain continuously employable.
- Work and excel in a highly competence supportive, multicultural, and professional environment which abiding to the legal and ethical responsibilities.

Practical – Course Outcome matrix

Course Outcomes (COs):

CO_3160707.1	Implement Networking and Data base connectivity in Java for given application.
CO_3160707.2	Implement webpage with dynamic content and server side web application using Servlet and JSP.
CO_3160707.3	Use web application framework JSF to build user interfaces.
CO_3160707.4	Use Object Relation Mapping using Hibernate to build database dependent applications
CO_3160707.5	Apply Model-View-Controller architecture to build complex client-server applications.

Sr. No.	**Objective(s) of Experiment**	**CO 1**	**CO 2**	**CO 3**	**CO 4**	**CO 5**
1.	Create databases related to your project and query them from console based java applications.	√				
2.	Create a client application that will connect with the application mentioned in the practical 1, client application will send a search “string” to the above application and above application will send back the string with the matching results after fetching from the database.	√				
3.	Modify practical 1 by replacing the console based user interface to a web based user interface. Use a servlet.		√			
4.	Modify the practical 4 to use JSP instead of Servlet.		√			
5.	Use JSF framework to replace JSP and calculate the reduction in programming			√		
6.	Make custom tag for a component that will be able to add/view/delete/modify Records			√		
7.	Use Object Relational Mapping and based on that prepare one configuration file along with the hibernet mapping file for 1 table of the application and test its working by replacing SQL to HQL.				√	
8.	Use Hibernet framework to replace JDBC calls and calculate the reduction in programming efforts for the entire application				√	
9.	Use Spring or any other MVC architecture and implement the Interface in that architecture that supports multi-tier architecture.					√
10.	Compare and analyze the JSF with the Spring framework.			√		√

Industry Relevant Skills

The following industry relevant competency are expected to be developed in the student by undertaking the practical work of this laboratory.

1. Will be able to develop web application using Netbeans/Ecllipse IDE
2. Will be able to use MVC architecture for application development
3. Will be able to use JSF and Hibernate framework

Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. We will have a project based learning methodology for practical in Advanced Java subject.
3. Students will form a team of maximum 4 people and then give the definition of a project..
4. Students can use the web application definition of Design Engineering subject also/ They can give another definition for a web application project
5. You'll need to decide the project and submit your practical list sample list is shown in next page.
6. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

Common Safety Instructions

Students are expected to

1) switch on the PC carefully (not to use wet hands)
2) shutdown the PC properly at the end of your Lab
3) carefully Handle the peripherals (Mouse, Keyboard, Network cable etc)
4) Use Laptop in lab after getting permission from Teacher

Sample Practical List

Project Name: Movies Review Website

Project Details: In the movie review website, the administrator will be able to enter the new movie details like movie_name, genre, language, movie_certificate details, release_date, etc. The administrator should be able to add, view, delete and modify movie details. The movie details and user details should have been stored in the database. Other users will be able to search the movie and fetch the details about a particular movie. They should be able to comment and review the movie. For review, the users will be entering a positive integer number between 0 to 10. For comment, they can write texts.

Sr. No.	Objective(s) of Experiment
1	In java console application, provide the user with following options: add movie, view movie, modify movie details, delete movie details. On entering a particular option on the console by user, perform appropriate operation and display the success/failure or any output on your console.
2	Make the practical 1 application as server for sending/receiving strings. Create a client application that will send a search movie string to the server application and server will replay with the matching movie name "string" back to the client application after fetching from the database.
3	Modify practical 1 by replacing the console based user interface to a web based user interface. Use a servlet.
4	Modify the practical 4 to use JSP instead of Servlet.
5	Use JSF framework to replace JSP and calculate the reduction in programming
6	Make custom tag for a component that will be able to add/view/delete/modify Records
7	Use Object Relational Mapping and based on that prepare one configuration file along with the hibernet mapping file for 1 table of the application and test its working by replacing SQL to HQL.
8	Use Hibernet framework to replace JDBC calls and calculate the reduction in programming efforts for the entire application
9	Use Spring or any other MVC architecture and implement the Interface in that architecture that supports multi-tier architecture.
10	Compare and analyze the JSF with the Spring/any other framework.

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Teacher with date	Remarks
1	In java console application, provide the user with following options: add movie, view movie, modify movie details, delete movie details. On entering a particular option on the console by user, perform appropriate operation and display the success/failure or any output on your console						
2	Make the practical 1 application as server for sending/receiving strings. Create a client application that will send a search movie string to the server application and server will replay with the matching movie name “string” back to the client application after fetching from the database.						
3	Modify practical 1 by replacing the console based user interface to a web based user interface. Use a servlet.						
4	Modify the practical 4 to use JSP instead of Servlet.						
5	Use JSF framework to replace JSP and calculate the reduction in programming						
6	Make custom tag for a component that will be able to add/view/delete/modify Records						
7	Use Object Relational Mapping and based on that prepare one configuration file along with the hibernet mapping file for 1 table of the application and test its working by replacing SQL to HQL.						
8	Use Hibernet framework to replace JDBC calls and calculate the reduction in programming efforts for the entire application						
9	Use Spring or any other MVC architecture and implement the Interface in that architecture that supports multi-tier architecture.						
10	Compare and analyze the JSF with the Spring/any other framework						

Experiment No: 1

In java console application, provide the user with following options: add movie, view movie, modify movie details, delete movie details. On entering a particular option on the console by user, perform appropriate operation and display the success/failure or any output on your console.

Date:

Competency and Practical Skills: programming and database commands

Relevant CO: CO1

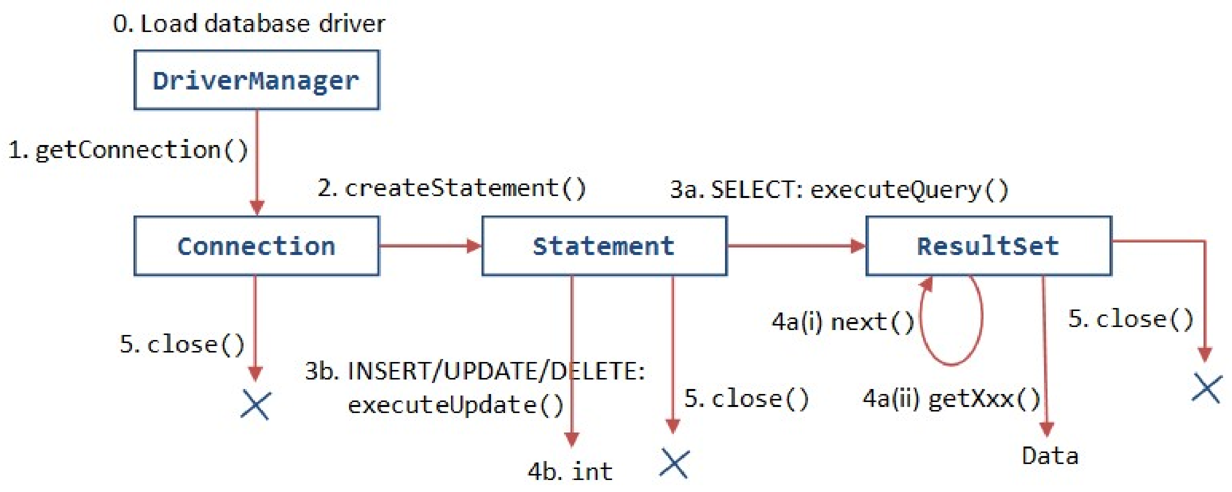
Objectives: (a) To show use of JDBC Drivers.
(b) To show how java program can interact with Database
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, Oracle Xpress edition 11g, Type4 JDBC driver for Oracle

Theory:

A JDBC program comprises the following FIVE steps:

- STEP 1: Allocate a `Connection` object, for connecting to the database server.
- STEP 2: Allocate a `Statement` object, under the `Connection` created earlier, for holding a SQL command.
- STEP 3: Write a SQL query and execute the query, via the `Statement` and `Connection` created.
- STEP 4: Process the query result.
- STEP 5: Close the `Statement` and `Connection` to free up the resources.



Safety and necessary Precautions:

1) Make sure the database server is started before running the program
2) Handle all necessary compile time Exception

Procedure:

1. Create a database for your project
2. Write java program to interact with tables in database using Type 4 JDBC driver.

Observations:

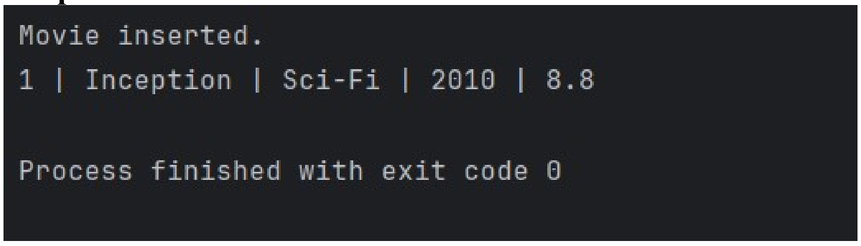
Create a database for your project
CREATE DATABASE MovieDB;

```
CREATE TABLE movies (
   movie_id INT PRIMARY KEY AUTO_INCREMENT,
   title VARCHAR(100),
   genre VARCHAR(50),
   release_year INT,
   rating FLOAT
);
```

Write java program to interact with tables in database using Type 4 JDBC driver

```
import java.sql. *;

public class MovieApp {
public static void main (String [] args) {
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         Connection con = DriverManager.getConnection(
            "jdbc: mysql://localhost:3306/MovieDB", "root", "");

         Statement stmt = con.createStatement();
         stmt.executeUpdate("INSERT INTO movies (title, genre, release_year, rating) VALUES
('Inception', 'Sci-Fi', 2010, 8.8)");
         System.out.println("Movie inserted.");

         ResultSet rs = stmt.executeQuery("SELECT * FROM movies");
         while (rs.next()) {
            System.out.println(rs.getInt(1) + " | " + rs.getString(2) + " | " +
                           rs.getString(3) + " | " + rs.getInt(4) + " | " + rs.getFloat(5));
         }

         con.close();
      } catch (Exception e) {
         System.out.println("Error: " + e);
      }
   }
}
```

Output:-



```
Movie inserted.
1 | Inception | Sci-Fi | 2010 | 8.8

Process finished with exit code 0
```

Suggested Reference:
JDBC™ API Tutorial and Reference, Third Edition, Maydene Fisher, Jon Ellis, Jonathan Bruce, Addison Wesley
References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 2

Make the practical 1 application as server for sending/receiving strings. Create a client application that will send a search movie string to the server application and server will replay with the matching movie name "string" back to the client application after fetching from the database.

Date:

Competency and Practical Skills: programming and database commands

Relevant CO: CO1

Objectives: (a) To show how to implement TCP/UDP communication between two java program using Socket/ ServerSocket and DatagramSocket/DatagramPacket.
(b) To show how to implement client server relationship between programs.
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse

Theory:

Java Networking Terminology: IP Address, Protocol, Port Number, MAC Address, Connection-oriented and connection-less protocol, Socket

1) IP Address: IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255. It is a logical address that can be changed.

2) Protocol: A protocol is a set of rules basically that is followed for communication. For example: TCP, FTP, Telnet, SMTP, POP etc.

3) Port Number: The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications. The port number is associated with the IP address for communication between two applications.

4) MAC Address: MAC (Media Access Control) address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC address. For example, an ethernet card may have a MAC address of 00:0d:83::b1:c0:8e.

5) Connection-oriented and connection-less protocol: In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP. But, in connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

6) Socket : A socket is an endpoint between two way communications.

java.net package

The java.net package can be divided into two sections: A Low-Level API: It deals with the abstractions of addresses i.e. networking identifiers, Sockets i.e. bidirectional data communication mechanism and Interfaces i.e. network interfaces .A High Level API: It deals with the abstraction of URIs i.e. Universal Resource Identifier, URLs i.e. Universal Resource Locator, and Connections i.e. connections to the resource pointed by URLs.

Safety and necessary Precautions:

1. Make sure the server/receiver program runs first
2. Handle all necessary compile time Exception

Procedure:

1. Create a Sender/Client program and Receiver/Server program.
2. Compile them
3. Run Server/Receiver program first and then run Client/Sender program

Observations:

ReceiverServer

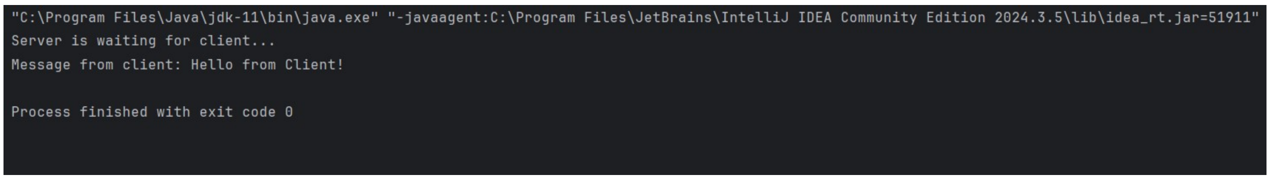
```
import java.net.*;
import java.io.*;

public class ReceiverServer {
   public static void main(String[] args) {
      try {
         ServerSocket server = new ServerSocket(5000);
         System.out.println("Server is waiting for client...");

         Socket socket = server.accept();
         BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         String message = in.readLine();

         System.out.println("Message from client: " + message);

         in.close();
         socket.close();
         server.close();
      } catch (Exception e) {
         System.out.println("Server Error: " + e.getMessage());
      }
   }
}
```

SenderClient

```
import java.net.*;
import java.io.*;

public class SenderClient {
   public static void main(String[] args) {
      try {
         Socket socket = new Socket("localhost", 5000);

         OutputStreamWriter out = new OutputStreamWriter(socket.getOutputStream());
         out.write("Hello from Client!\n");
         out.flush();

         out.close();
         socket.close();
      } catch (Exception e) {
            System.out.println("Client Error: " + e.getMessage());
        }
    }
}
```

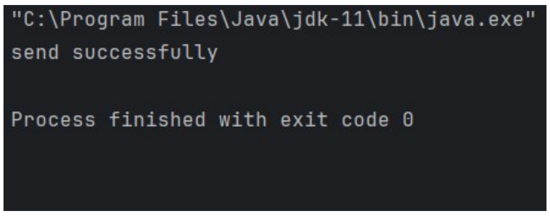
Output:-

ReceiverServer



```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=51911"
Server is waiting for client...
Message from client: Hello from Client!

Process finished with exit code 0
```

senderclient



```
"C:\Program Files\Java\jdk-11\bin\java.exe"
send successfully

Process finished with exit code 0
```

Suggested Reference: Professional Java Server Programming by Subrahmanyam Allamaraju, Cedric Buest Wiley Publication

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 3

Modify practical 1 by replacing the console based user interface to a web based user interface. Use a servlet.

Date:

Competency and Practical Skills: Design a web application using Servlet

Relevant CO: CO2

Objectives: (a) implement webpages using HTML for collecting user input.
(b) able to do server side programming using Servlet to process user input.
(c) To show how Servlet can interact with Database
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Apache Tomcat Server, Netbeans/eclipse, Oracle Xpress edition 11g, Type4 JDBC driver for Oracle

Theory:

- Servlet technology is used to create web application
- Servlet technology is robust and scalable because of java language.
- Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language.
- There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

Servlet APIs

- javax.servlet and javax.servlet.http
- javax.servlet package contains many interfaces and classes that are used by the servlet or web container.
- javax.servlet.http package contains interfaces and classes that are responsible for http requests

The servlet can be created by three ways:

By implementing Servlet interface,
By inheriting GenericServlet class,
By inheriting HttpServlet class

- The mostly used approach is by extending HttpServlet.

The steps are as follows:

- Create a directory structure
- Create a Servlet
- Compile the Servlet
- Create a deployment descriptor
- Start the server(Apache tomcat) and deploy the project
- Access the servlet

Safety and necessary Precautions:

1. Make sure the database server and application server are started before running the program
2. Put you application directory in to root directory of Tomcat server
3. To run the program use any web browser
4. Handle all necessary compile time Exception

Procedure:

1. Create a directory structure for web application in application folder.
2. Create database for your project
3. Create an html file asking for data from user and submit those data to servlet in application folder
4. Create a servlet class to receive the input and process it as per your requirement, Also this servlet program can interact with database using Type 4 JDBC driver.
5. Prepare web.xml file to put deployment descriptor
6. Compile to servlet program
7. Put application folder into root directory of application server(wepapps folder of Tomcat server)
8. Open any web browser and run the program by typing url in address bar.

Observations:

java

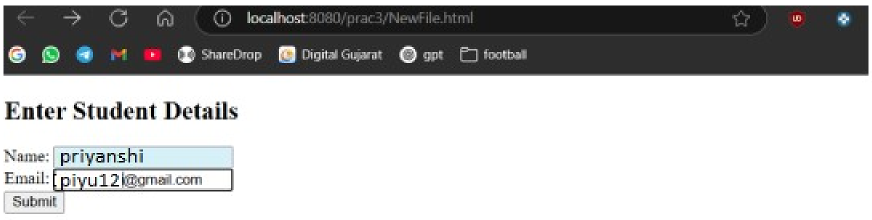
```
package prac3;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@SuppressWarnings("serial")
public class prac3a extends HttpServlet {
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

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

            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                Connection con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/StudentDB", "root", "");

                PreparedStatement ps = con.prepareStatement(
                    "INSERT INTO student (name, email) VALUES (?, ?)");

                ps.setString(1, name);
                ps.setString(2, email);
                ps.executeUpdate();

                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<h3>Data Saved Successfully!</h3>");
          }
}
```

Html

```
<!DOCTYPE html>
<html>
<head><title>Student Form</title></head>
<body>
   <h2>Enter Student Details</h2>
   <form action="prac3a" method="post">
      Name: <input type="text" name="name"><br>
      Email: <input type="email" name="email"><br>
      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

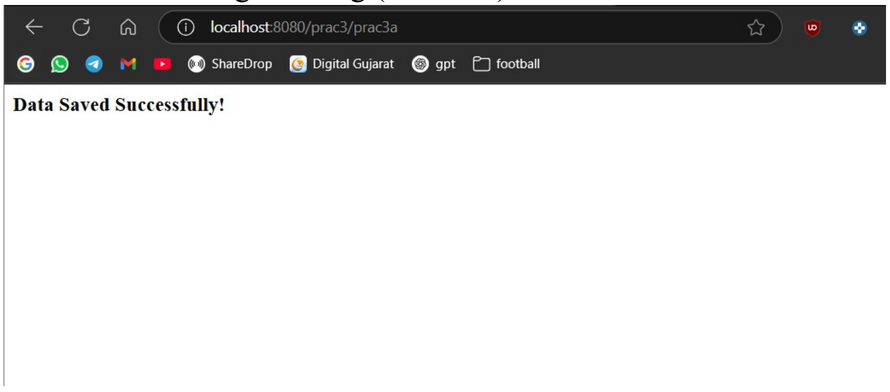
Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
       https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
       version="6.0">

  <servlet>
    <servlet-name>prac3a</servlet-name>
    <servlet-class>prac3.prac3a</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>prac3a</servlet-name>
    <url-pattern>/prac3a</url-pattern>
  </servlet-mapping>

</web-app>
```



Enter Student Details

Name: priyanshi
Email: piyu12@gmail.com
Submit



Data Saved Successfully!

.

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 4

Modify the practical 3 to use JSP instead of Servlet.

Date:

Competency and Practical Skills: Design a web application using JSP

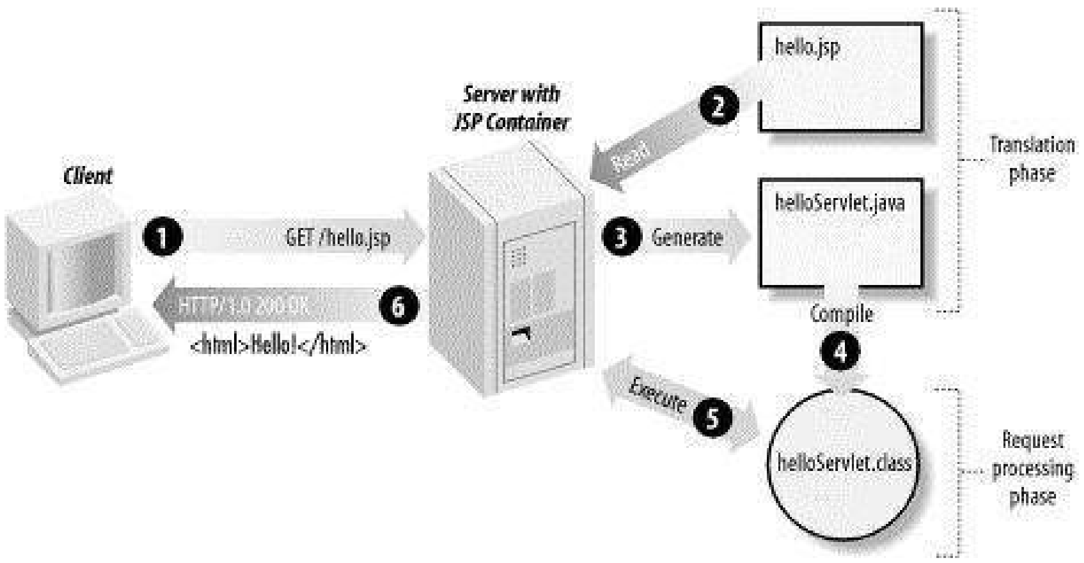
Relevant CO: CO2

Objectives: (a) implement webpages using HTML for collecting user input.
(b) able to do server side programming using JSP to process user input.
(c) To show how JSP can interact with Database
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Apache Tomcat Server, Netbeans/eclipse, Oracle Xpress edition 11g, Type4 JDBC driver for Oracle

Theory:
Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.

JSP can be run as per the following diagram –



Steps to configure JDK and Tomcat server for servlet and JSP program

1) Download Java (JDK) and install it.

2) Set PATH and JAVA_HOME environment variable

set PATH = C:\jdk1.5.0_20\bin;%PATH%
set JAVA_HOME = C:\jdk1.5.0_20
(Assuming java installed in c:\ and jdk1.5.0_20)

3) Download the latest version of Tomcat and install it

4) **Create a CATALINA_HOME** environment variable pointing to **C:\apache-tomcat-5.5.29**. (assuming tomcat is installed in **C:\apache-tomcat-5.5.29)**

5) Run following command to start tomcat server

C:\apache-tomcat-5.5.29\bin\startup.bat
After a successful startup, the default web-applications included with Tomcat will be available by visiting **http://localhost:8080/**.

6) Tomcat can be stopped by executing the following commands on the Windows machine –

C:\apache-tomcat-5.5.29\bin\shutdown

7) Set classpath as below,
set CATALINA = C:\apache-tomcat-5.5.29
set CLASSPATH = %CATALINA%\common\lib\jsp-api.jar;%CLASSPATH%

Safety and necessary Precautions:

1. Make sure the database server and application server are started before running the program
2. Put you application directory in to root directory of Tomcat server
3. To run the program use any web browser
4. Handle all necessary compile time Exception

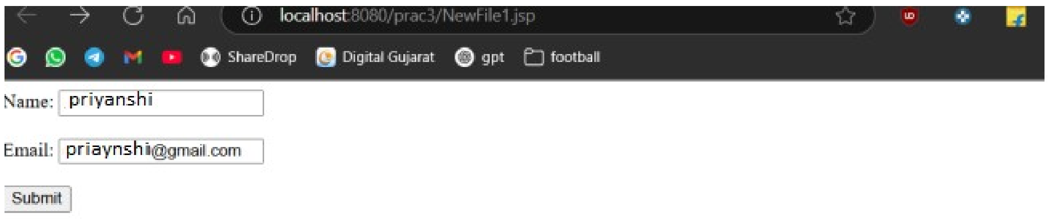
Procedure:

1. Create a directory structure for web application in application folder.
2. Create database for your project
3. Create an html file asking for data from user and submit those data to jsp in application folder
4. Create a jsp program to receive the input and process it as per your requirement, Also this jsp program can interact with database using Type 4 JDBC driver.
5. Prepare web.xml file to put deployment descriptor
6. Put application folder into root directory of application server(wepapps folder of Tomcat server)
7. Open any web browser and run the program by typing url in address bar.

Observations:

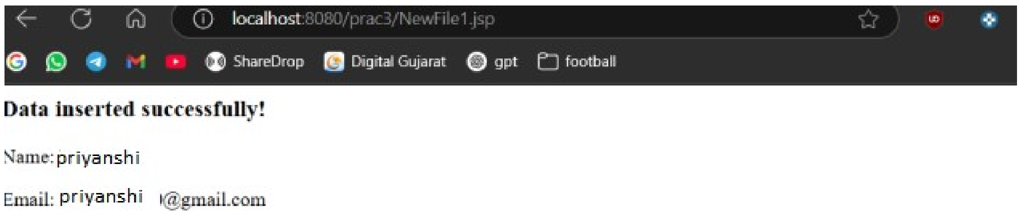
```
<%@ page import="java.sql.*" %>
<html>
<head>
   <title>Merge HTML and JSP</title>
</head>
<body>

<% if(request.getParameter("submit") == null) { %>
   <!-- Show the form -->
   <form method="post" action="">
      Name: <input type="text" name="name" required><br><br>
      Email: <input type="email" name="email" required><br><br>
      <input type="submit" name="submit" value="Submit">
   </form>
<% } else {
   String name = request.getParameter("name");
   String email = request.getParameter("email");

   try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection con = DriverManager.getConnection(
         "jdbc:mysql://localhost:3306/StudentDB", "root", "");

      String sql = "INSERT INTO Student (name, email) VALUES (?, ?)";
      PreparedStatement ps = con.prepareStatement(sql);
      ps.setString(1, name);
      ps.setString(2, email);
      ps.executeUpdate();

      out.println("<h3>Data inserted successfully!</h3>");
      out.println("<p>Name: " + name + "</p>");
      out.println("<p>Email: " + email + "</p>");
```

```
        ps.close();
        con.close();
    } catch(Exception e) {
        out.println("Error: " + e.getMessage());
    }
} %>

</body>
</html>
```



Name: priyanshi

Email: priaynshi@gmail.com



Data inserted successfully!

Name: priyanshi

Email: priyanshi @gmail.com

Suggested Reference:
Complete Reference J2EE by James Keogh mcgraw publication
Professional Java Server Programming by Subrahmanyam Allamaraju, Cedric Buest Wiley Publication

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 5

Use JSF framework to replace JSP and calculate the reduction in programming.

Date:

Competency and Practical Skills: Design a web application using JSF Framework

Relevant CO: CO3

Objectives: (a) implement dynamic web-application using JSF framework.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Apache Tomcat Server, Netbeans/eclipse, Oracle Xpress edition 11g, Type4 JDBC driver for Oracle, JSF2.2 jar file

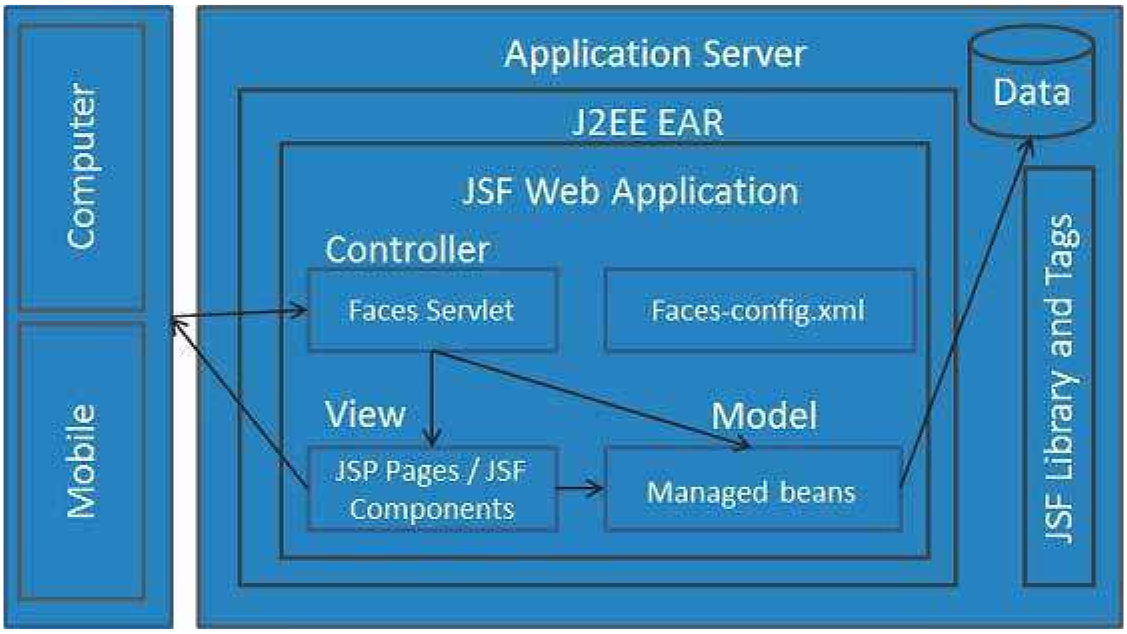
Theory:

Java Server Faces (JSF) is a Java-based web application framework intended to simplify development integration of web-based user interfaces.
Basic understanding of Java programming language, and understanding of other web technologies such as HTML, CSS, AJAX, etc. is required to use JSF.

JSF facilitates Web application development by –

- Providing reusable UI components
- Making easy data transfer between UI components
- Managing UI state across multiple server requests
- Enabling implementation of custom components
- Wiring client-side event to server-side application code

JSF Architecture



A JSF application consists of web pages with JSF UI components. A JSF application requires also some configuration files ("faces-config.xml" and web.xml).

The faces-config.xml defines:

- Managed Bean - the data elements of the JSF application (managed beans and backing beans) represent a Java class which will be created dynamically during runtime of the JSF application. It can be defined for which scope the bean is valid (Session, Request, Application or none)
- the navigation between web pages

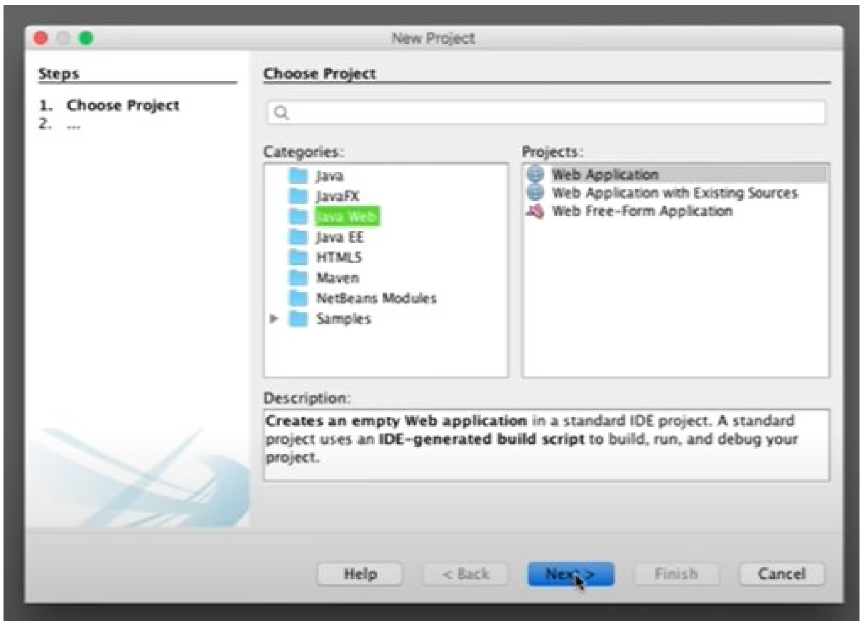
- data validators - Used to check the validity of UI input
- data converters -Used to translate between UI and model

Safety and necessary Precautions:

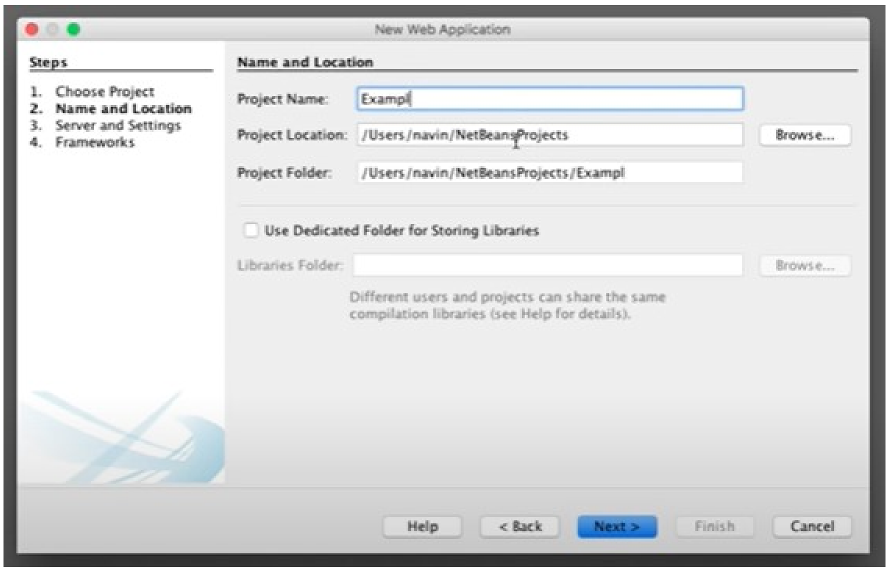
1. Make sure the database server and application server are started before running the program
2. Put you application directory in to root directory of Tomcat server
3. To run the program use any web browser
4. Handle all necessary compile time Exception

Procedure:

1) Create database for your project
2) Open netbeans and create a new project (web application with JSF)



Description:
Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.



Project Name: Exampl

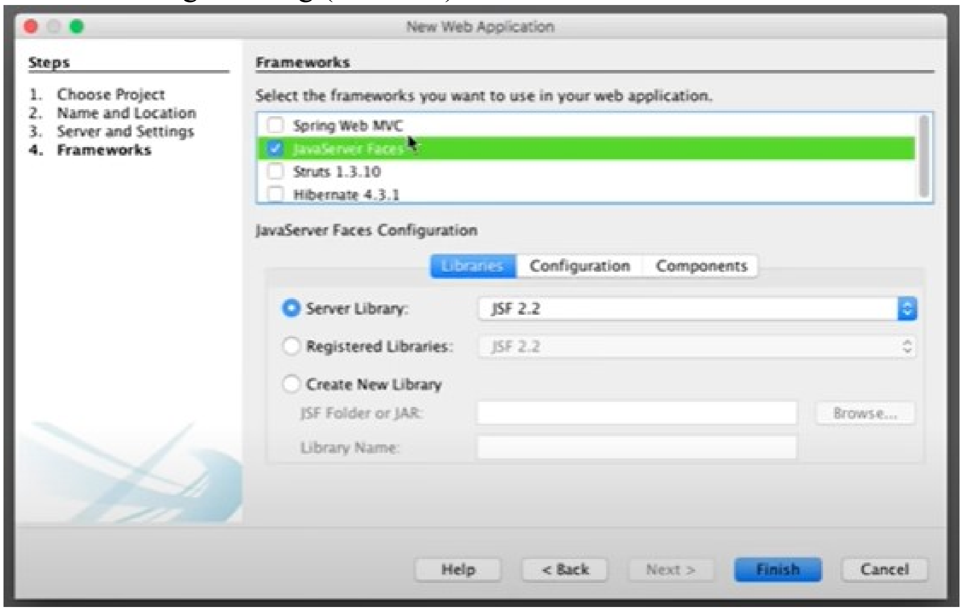
Project Location: /Users/navin/NetBeansProjects

Project Folder: /Users/navin/NetBeansProjects/Exampl

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).



Frameworks

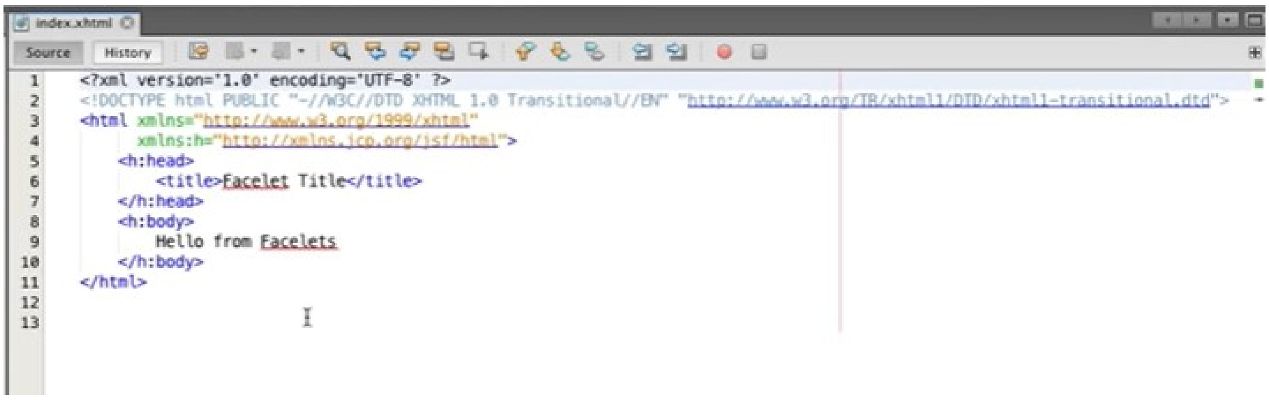
Select the frameworks you want to use in your web application.

- [ ] Spring Web MVC
- [✓] JavaServer Faces
- [ ] Struts 1.3.10
- [ ] Hibernate 4.3.1

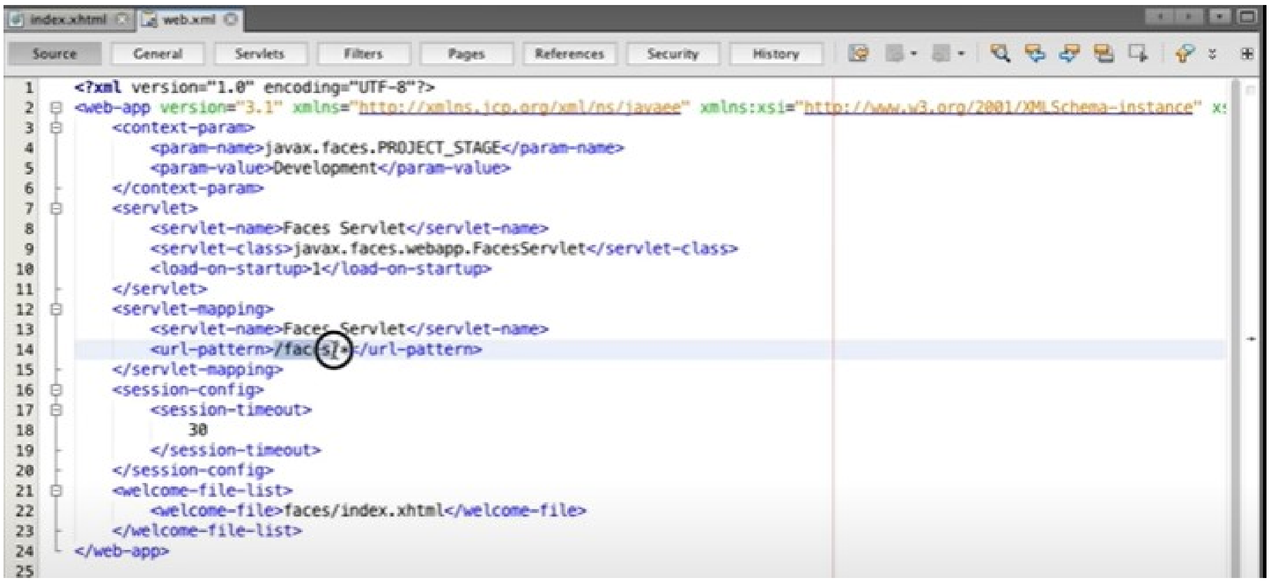
JavaServer Faces Configuration

Libraries | Configuration | Components

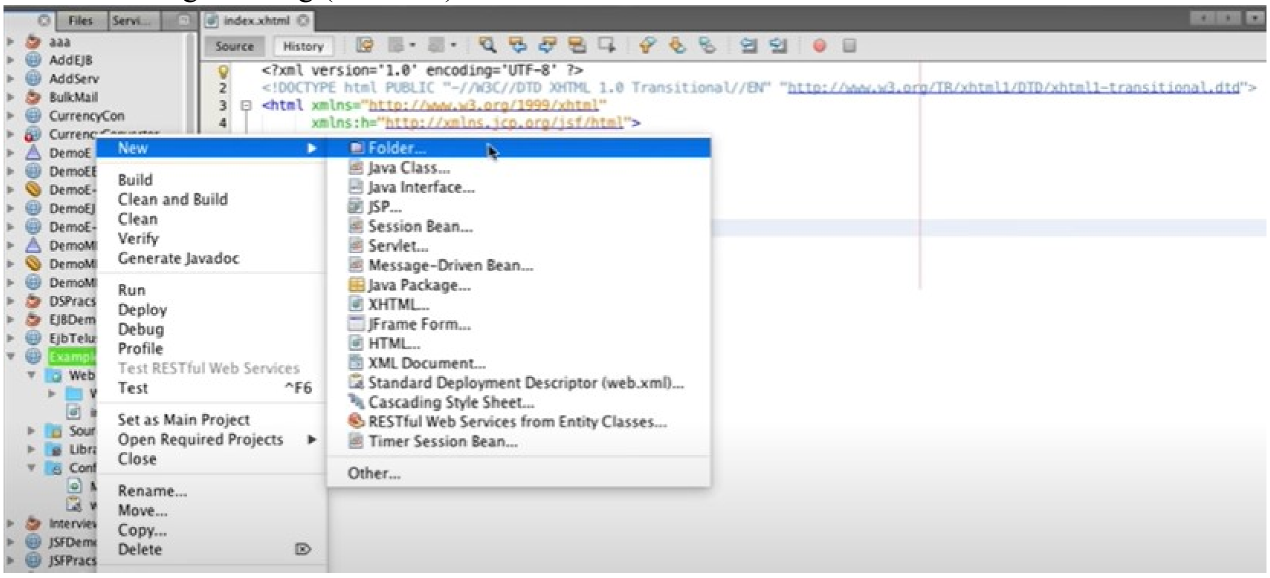
- [✓ Server Library:] JSF 2.2
- Registered Libraries: JSF 2.2
- Create New Library
  - JSF Folder or JAR:
  - Library Name:



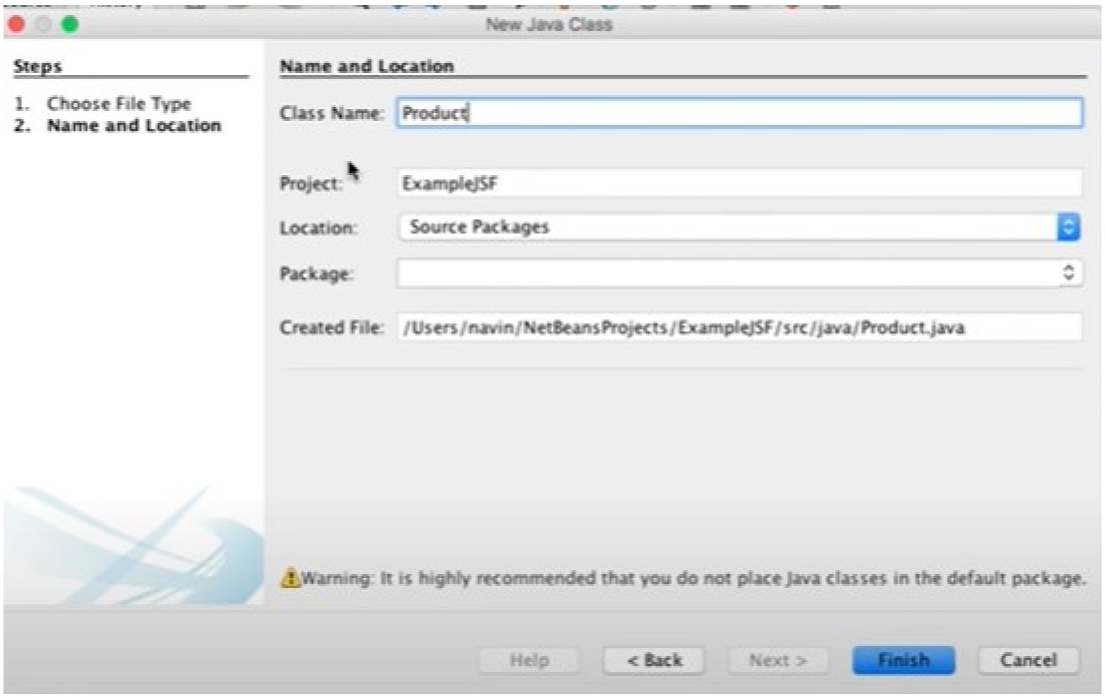
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        Hello from Facelets
    </h:body>
</html>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
```

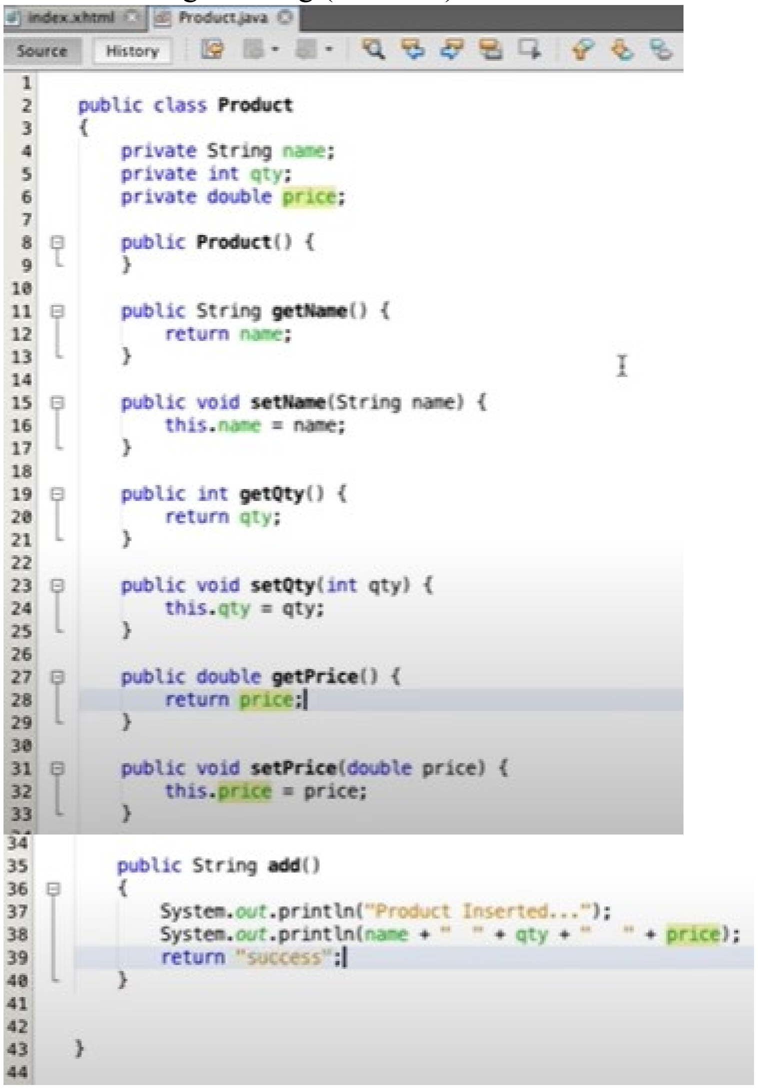

New Java Class

Steps
1. Choose File Type
2. Name and Location

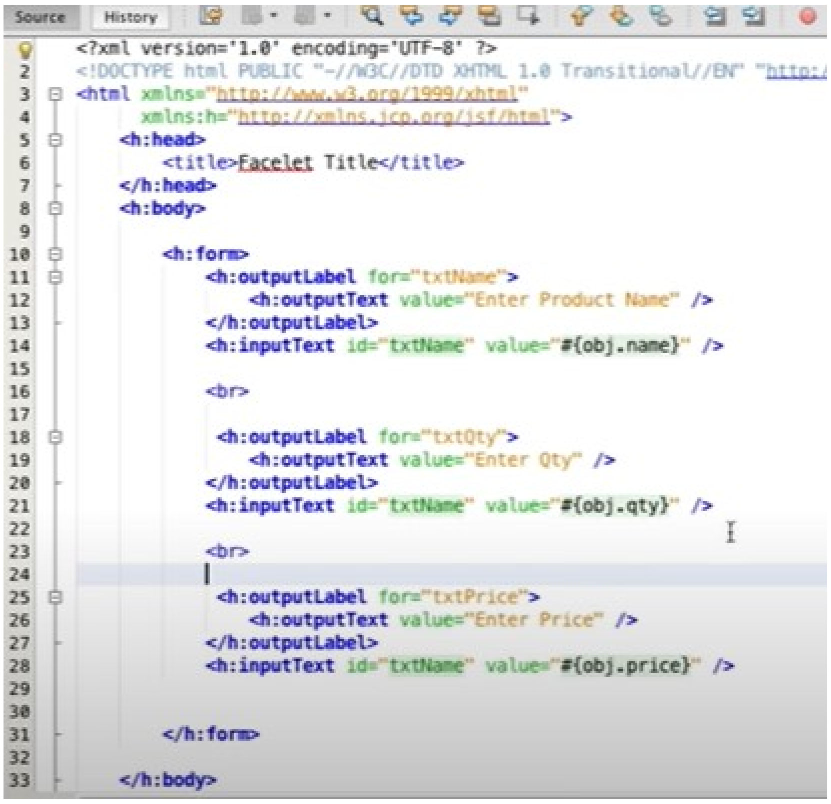
Name and Location

Class Name: Product
Project: ExampleJSF
Location: Source Packages
Package:
Created File: /Users/navin/NetBeansProjects/ExampleJSF/src/java/Product.java

Warning: It is highly recommended that you do not place Java classes in the default package.



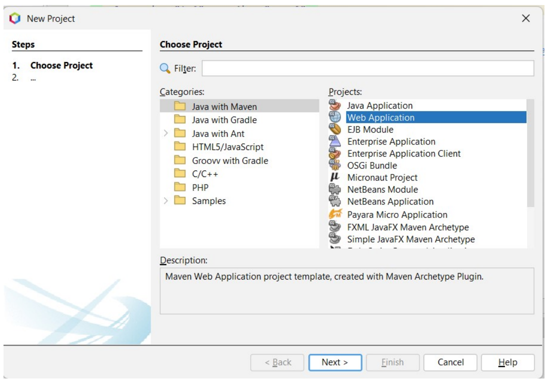
```java

public class Product
{
    private String name;
    private int qty;
    private double price;

    public Product() {
    }

    public String getName() {
        return name;
    }

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

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

    public double getPrice() {
        return price;
    }

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

    public String add()
    {
        System.out.println("Product Inserted...");
        System.out.println(name + "    " + qty + "    " + price);
        return "success";
    }


}

```



```xml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http:/
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>

        <h:form>
            <h:outputLabel for="txtName">
                <h:outputText value="Enter Product Name" />
            </h:outputLabel>
            <h:inputText id="txtName" value="#{obj.name}" />

            <br>

             <h:outputLabel for="txtQty">
                <h:outputText value="Enter Qty" />
            </h:outputLabel>
            <h:inputText id="txtName" value="#{obj.qty}" />

            <br>

             <h:outputLabel for="txtPrice">
                <h:outputText value="Enter Price" />
            </h:outputLabel>
            <h:inputText id="txtName" value="#{obj.price}" />


        </h:form>

    </h:body>
```

3) Add or modify the code in files mentioned in above screenshot
4) Run the project.

Observations:

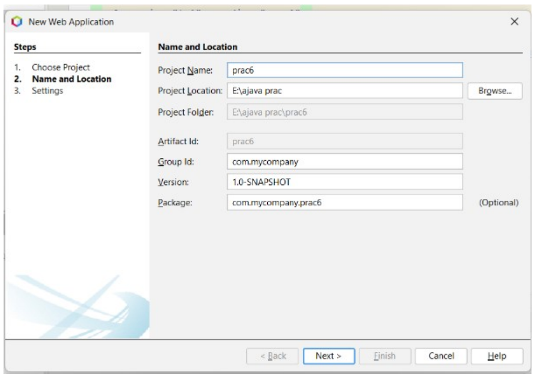

New Project

Steps
1. Choose Project
2. ...

Choose Project
Filter:

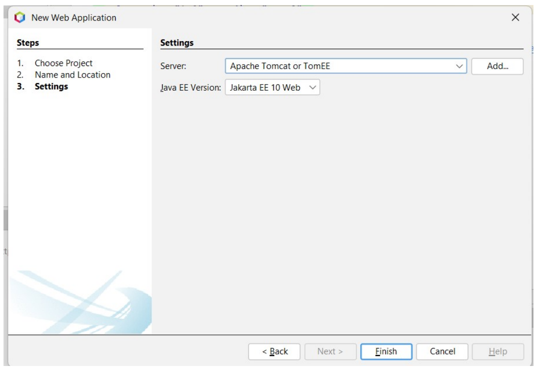
Categories:	Projects:
Java with Maven	Java Application
Java with Gradle	Web Application
Java with Ant	EJB Module
HTML5/JavaScript	Enterprise Application
Groovy with Gradle	Enterprise Application Client
C/C++	OSGi Bundle
PHP	Micronaut Project
Samples	NetBeans Module
	NetBeans Application
	Payara Micro Application
	FXML JavaFX Maven Archetype
	Simple JavaFX Maven Archetype

Description:
Maven Web Application project template, created with Maven Archetype Plugin.


New Web Application

Steps
1. Choose Project
2. Name and Location
3. Settings

Name and Location
Project Name: prac6
Project Location: E:\ajava prac
Project Folder: E:\ajava prac\prac6
Artifact Id: prac6
Group Id: com.mycompany
Version: 1.0-SNAPSHOT
Package: com.mycompany.prac6 (Optional)


New Web Application

Steps
1. Choose Project
2. Name and Location
3. Settings

Settings
Server: Apache Tomcat or TomEE
Java EE Version: Jakarta EE 10 Web

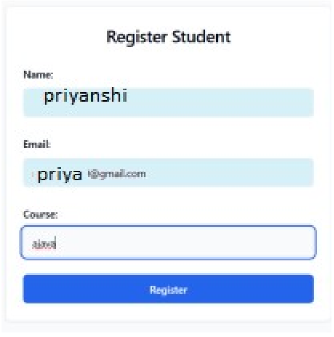
```
package model;

import javax.faces.bean.ManagedBean;

@ManagedBean
public class Student {
    private String name;
    private String email;
    private String course;

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

    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }

    public String getCourse() {
        return course;
    }
    public void setCourse(String course) {
        this.course = course;
    }

    public String register() {
        return "success";  // redirects to success.xhtml
    }
}
```

index.xhtml

```
<html>
<h:head>
    <title>Student Registration</title>
</h:head>
<h:body>
    <h:form>
        <h3>Register Student</h3>

        <h:outputLabel value="Name:" />
        <h:inputText value="#{student.name}" /><br/><br/>

        <h:outputLabel value="Email:" />
        <h:inputText value="#{student.email}" /><br/><br/>

        <h:outputLabel value="Course:" />
        <h:inputText value="#{student.course}" /><br/><br/>

        <h:commandButton value="Register" action="#{student.register}" />
    </h:form>
</h:body>
</html>
```

Success.xhtml

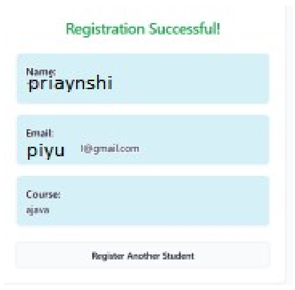
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head>
    <title>Success</title>
</h:head>
<h:body>
    <h2>Registration Successful!</h2>
</h:body>
</html>
```


Register Student

Name:
priyanshi

Email:
priya l@gmail.com

Course:
ajava


Registration Successful!

Name:
priaynshi

Email:
piyu l@gmail.com

Course:
ajava

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 6

Make custom tag for a component that will be able to add/view/delete/modifyRecords

Date:

Competency and PracticalSkills:programming and database commands

Relevant CO:CO3

Objectives: (a) To be able to use custom tag for use in JSP for converting java scriptlets to tags in a JSP page.
(b) To be able to implement the custom tag for CRUD operations.
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, Oracle Xpress edition 11g, Apache Tomcat / GlassFish / any other web server for Java.

Theory:
Custom tags are user-defined tags. They eliminates the possibility of scriptlet tag and separates the business logic from the JSP page.

The same business logic can be used many times by the use of custom tag.

Advantages of Custom Tags
The key advantages of Custom tags are as follows:

Eliminates the need of scriptlet tag The custom tags eliminates the need of scriptlet tag which is considered bad programming approach in JSP.
Separation of business logic from JSP The custom tags separate the the business logic from the JSP page so that it may be easy to maintain.
Re-usability The custom tags makes the possibility to reuse the same business logic again and again.

Safety and necessary Precautions:

3) Make sure the database server is started before running the program
4) Make sure the Web Server is started before calling the jsp page.
5) Handle all necessary compile time Exception

Procedure:

1. **Create the Tag handler class and perform action at the start or at the end of the tag.**
   To create the Tag Handler, we are inheriting the TagSupport class and overriding its method doStartTag().To write data for the jsp, we need to use the JspWriter class.The PageContext class provides getOut() method that returns the instance of JspWriter class. TagSupport class provides instance of pageContextbydefault.
2. **Create the Tag Library Descriptor (TLD) file and define tags**
   Tag Library Descriptor (TLD) file contains information of tag and Tag Hander classes. It must be contained inside the WEB-INF directory.
3. **Create the JSP file that uses the Custom tag defined in the TLD file**
   Let's use the tag in our jsp file. Here, we are specifying the path of tld file directly. But it is recommended to use the uri name instead of full path of tld file. We will learn about urilater.It uses taglib directive to use the tags defined in the tld file.

Observations:

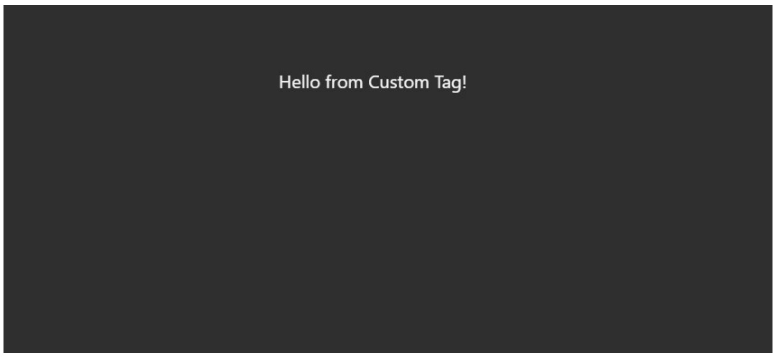
```
package customtags;

import jakarta.servlet.jsp.tagext.TagSupport;
import jakarta.servlet.jsp.JspWriter;
import jakarta.servlet.jsp.JspException;
import java.io.IOException;

public class HelloTag extends TagSupport {
    @Override
    public int doStartTag() throws JspException {
        try {
            JspWriter out = pageContext.getOut();
            out.print("Hello from Custom Tag!");
        } catch (IOException e) {
            throw new JspException("Error in HelloTag", e);
        }
        return SKIP_BODY;
    }
}
```

TLD

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <short-name>h</short-name>
  <uri>/mytags</uri>
  <tag>
    <name>sayhello</name>
    <tag-class>customtags.HelloTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

Jsp

```
<%@ taglib prefix="h" uri="/mytags" %>

<html>
  <body>
    <h:sayhello />
  </body>
</html>
```



```
Hello from Custom Tag!
```

```
    version="2.0">

  <tlib-version>1.0</tlib-version>
  <short-name>CustomTags</short-name>
  <uri>/WEB-INF/customtags</uri>

  <tag>
    <name>myTag</name>
    <tag-class>com.example.MyTagHandler</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>message</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

Suggested Reference:
https://docs.oracle.com/cloud/latest/as111170/TAGLB/quickstart.html
Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 7

Use Object Relational Mapping and based on that prepare one configuration file along with the hibernet mapping file for 1 table of the application and test its working by replacing SQL to HQL.

Date:

Competency and PracticalSkills: programming and database commands

Relevant CO: CO4

Objectives: (a) To understand object relational impedance and prepare the object relational mapping and mapping file
(b) To understand and prepare hibernet configuration file
(c) To use HQL to execute a query
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, Hibernetframwork

Theory:

Hibernate Framework

Following are the advantages of hibernate framework:

1) Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

2) Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

Elements of Hibernate Architecture

For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows:

SessionFactory

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Session

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

Transaction

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

ConnectionProvider

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

TransactionFactory

It is a factory of Transaction. It is optional.

Safety and necessary Precautions:

3. Make sure the database server is up and running.
4. Make sure all the hibernet libraries are put in the classpath
5. Make sure the web server is up and running
6. Handle all necessary compile time Exception

Procedure:

For creating the first hibernate application, we need to follow the following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object
5. Load the jar file
6. Run the first hibernate application by using command prompt

Observations:

hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/testdb</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password"></property>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>

    <mapping resource="student.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Student.java

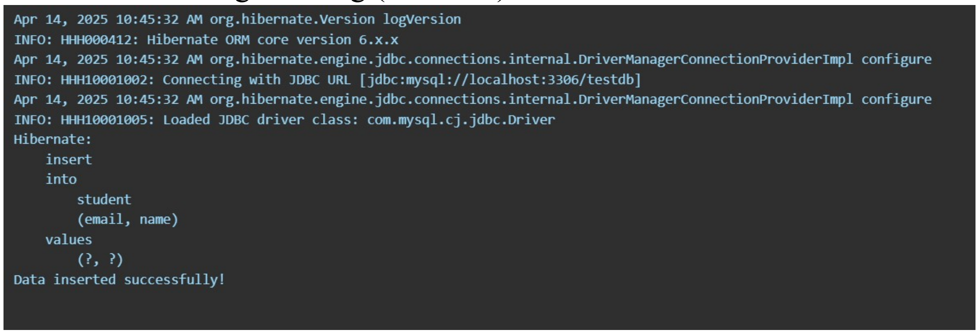
```
package com.demo;

public class Student {
    private int id;
    private String name;
    private String email;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
```

Student.hbm.xml

```
<hibernate-mapping>
  <class name="com.demo.Student" table="student">
    <id name="id" column="id">
      <generator class="native"/>
    </id>
    <property name="name"/>
    <property name="email"/>
  </class>
</hibernate-mapping>
```

Main class

```
package com.demo;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class InsertData {
    public static void main(String[] args) {
        Student s = new Student();
        s.setName("John Doe");
        s.setEmail("john@example.com");

        SessionFactory factory = new Configuration().configure().buildSessionFactory();
        Session session = factory.openSession();

        session.beginTransaction();
        session.save(s);
        session.getTransaction().commit();

        session.close();
        factory.close();

        System.out.println("Data inserted successfully!");
    }
}
```



```
Apr 14, 2025 10:45:32 AM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate ORM core version 6.x.x
Apr 14, 2025 10:45:32 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH10001002: Connecting with JDBC URL [jdbc:mysql://localhost:3306/testdb]
Apr 14, 2025 10:45:32 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH10001005: Loaded JDBC driver class: com.mysql.cj.jdbc.Driver
Hibernate:
    insert
    into
        student
        (email, name)
    values
        (?, ?)
Data inserted successfully!
```

Suggested Reference:
https://docs.oracle.com/cd/E11035_01/workshop102/ormworkbench/hibernate-tutorial/tutHibernate1.html

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 8

Use Hibernet framework to replace JDBC calls and calculate the reduction in programming efforts for the entire application

Date:

Competency and PracticalSkills: programming and database commands

Relevant CO: CO3

Objectives: (a) To be able implement the hibernet concepts in a project.
(b) To be able to implement the hibernet and HQL for complex SQL queries and CRUD operations.
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, Oracle Xpress edition 11g, Apache Tomcat / GlassFish / any other web server for Java.Hibernet framework.

Theory:
Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL. So it is database independent query language.

Advantage of HQL
There are many advantages of HQL. They are as follows:
1. database independent
2. supports polymorphic queries
3. easy to learn for Java Programmer

Query Interface
It is an object oriented representation of Hibernate Query. The object of Query can be obtained by calling the createQuery() method Session interface.

The query interface provides many methods. There is given commonly used methods:
Play Video

- publicintexecuteUpdate() is used to execute the update or delete query.
- public List list() returns the result of the ralation as a list.
- public Query setFirstResult(introwno) specifies the row number from where record will be retrieved.
- public Query setMaxResult(introwno) specifies the no. of records to be retrieved from the relation (table).
- public Query setParameter(int position, Object value) it sets the value to the JDBC style query parameter.
- public Query setParameter(String name, Object value) it sets the value to a named query parameter.

Procedure:
For creating the first hibernate application in Eclipse IDE, we need to follow the following steps:

1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file

6. Create the class that retrieves or stores the persistent object
7. Run the application

Observations:

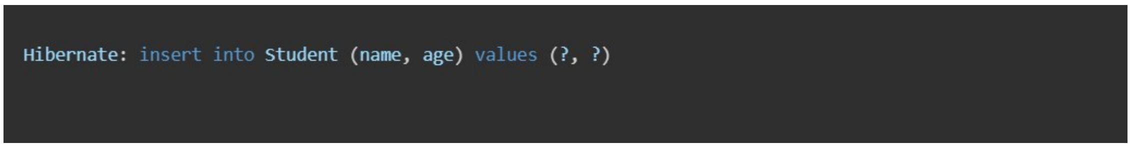
```
import javax.persistence.*;

@Entity
@Table(name = "Student")
public class Student {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;
   private String name;
   private int age;
}
```

hibernate.cfg.xml

```
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/your_database</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <mapping class="Student"/>
    </session-factory>
</hibernate-configuration>
```

Main.java

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class StudentDemo {
    public static void main(String[] args) {
        SessionFactory factory = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Student.class).buildSessionFa
ctory();
        Session session = factory.getCurrentSession();

        try {
            // Create a new student object
            Student student = new Student("John", 22);

            session.beginTransaction();
            session.save(student);
            session.getTransaction().commit();
        } finally {
            factory.close();
        }
    }
}
```



```
Hibernate: insert into Student (name, age) values (?, ?)
```

Conclusion:
creating the first Hibernate application in Eclipse involves setting up the project with required JARs, defining the persistent class, configuration, and running the main class to perform database operations.

Suggested Reference:
https://docs.oracle.com/cloud/latest/as111170/TAGLB/quickstart.htm

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											

Experiment No: 9

Use Spring or any other MVC architecture and implement the Interface in that architecture that supports multi-tier architecture.
Date:

Competency and Practical Skills: programming and database commands

Relevant CO: CO5

Objectives: (a) To understand MVC architecture.
(b) To understand and use Spring as a reference MVC architecture for ease of programming
(c) To assess the impact of using a framework instead of pure JSP file for development, maintenance etc.
.
Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, Spring and Hibernetframwork

Theory:

Spring is a lightweight framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

Inversion Of Control (IOC) and Dependency Injection
These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain. Let's understand this with the following code:

```
class Employee{
Address address;
Employee(){
address=new Address();
}
}
```

In such case, there is dependency between the Employee and Address (tight coupling). In the Inversion of Control scenario, we do this something like this:

```
class Employee{
Address address;
Employee(Address address){
this.address=address;
}
}
```

Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

Advantage of Dependency Injection

- makes the code loosely coupled so easy to maintain
- makes the code easy to test

Advantages of Spring Framework

There are many advantages of Spring Framework. They are as follows:

1) Predefined Templates

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

2) Loose Coupling

The Spring applications are loosely coupled because of dependency injection.

3) Easy to test

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

4) Lightweight

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

5) Fast Development

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

6) Powerful abstraction

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

7) Declarative support

It provides declarative support for caching, validation, transactions and formatting.

Safety and necessary Precautions:

1. Make sure the database server is up and running.
2. Make sure all the hibernet and spring libraries are put in the classpath
3. Make sure the web server is up and running
4. Handle all necessary compile time Exception

Procedure:

Spring MVC CRUD Example

CRUD (Create, Read, Update and Delete) application is the most important application for creating any project. It provides an idea to develop a large project. In spring MVC, we can develop a simple CRUD application.

Here, we are using JdbcTemplate for database interaction.

- Create a table

1. Add dependencies to pom.xml file.
2. Create the bean class
3. Create the controller class
4. Create the DAO class
5. Provide the entry of controller in the web.xml file
6. Define the bean in the xml file spring-servlet.xml
7. Create the requested page
8. Create the other view components

Observations:

Bean Class

```
public class Student {
   private int id;
   private String name;
   private String email;
}
```

StudentDao.java

```
public class StudentDAOImpl implements StudentDAO {
   JdbcTemplate template;

   public void setTemplate(JdbcTemplate template) {
      this.template = template;
   }
   public void save(Student s) {
      String sql = "INSERT INTO student(name,email) VALUES(?,?)";
      template.update(sql, s.getName(), s.getEmail());
   }
   public List<Student> getAll() {
      return template.query("SELECT * FROM student", new
BeanPropertyRowMapper<>(Student.class));
   }
   public void delete(int id) {
      template.update("DELETE FROM student WHERE id=?", id);
   }
}
```

StudentController.class

```
@Controller
public class StudentController {
   @Autowired
   StudentDAO dao;
   @RequestMapping("/save")
   public String save(@ModelAttribute Student s) {
      dao.save(s);
      return "redirect:/view";
   }
   @RequestMapping("/view")
   public ModelAndView view() {
      List<Student> list = dao.getAll();
```

```
        return new ModelAndView("view", "list", list);
    }
    @RequestMapping("/delete")
     public String delete(@RequestParam int id) {
        dao.delete(id);
        return "redirect:/view";
    }
}
```

spring-servlet.xml

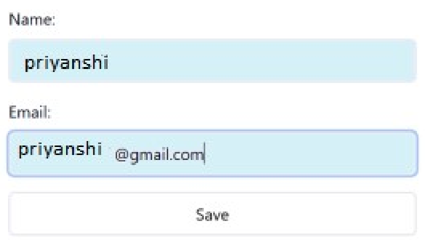
```
<beans>
  <context:component-scan base-package="com.example" />
  <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
  </bean>

  <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/test" />
    <property name="username" value="root" />
    <property name="password" value="root" />
  </bean>

  <bean id="template" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="ds" />
  </bean>

  <bean id="dao" class="com.example.StudentDAOImpl">
    <property name="template" ref="template" />
  </bean>
</beans>
```

save.jsp

```
<form action="save" method="post">
    Name: <input type="text" name="name"/><br/>
    Email: <input type="text" name="email"/><br/>
    <input type="submit" value="Save"/>
</form>
```

view.jsp

```
<table border="1">
<tr><th>ID</th><th>Name</th><th>Email</th><th>Action</th></tr>
<c:forEach var="s" items="${list}">
<tr>
    <td>${s.id}</td>
    <td>${s.name}</td>
    <td>${s.email}</td>
    <td><a href="delete?id=${s.id}">Delete</a></td>
</tr>
</c:forEach>
</table>
```

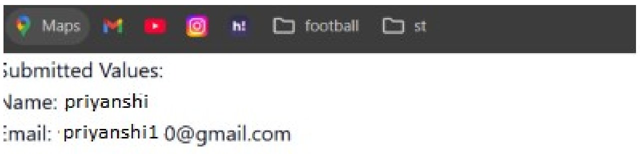


Name:

priyanshi

Email:

priyanshi @gmail.com



```
ubmitted Values:
lame: priyanshi
:mail: priyanshi1 0@gmail.com
```

Suggested Reference:
https://spring.io/

References used by the students:

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition (2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Average (1)**	**Good (2)**	**Average (1)**	**Good (2)**	**Average (1)**	**Good (2)**	**Average (1)**	**Good (2)**	**Average (1)**	
Marks											

Experiment No: 10

Compare and analyze the JSF with the Spring framework. Use JSF for creating a CRUD operation.

Date:

Competency and PracticalSkills: programming and database commands

Relevant CO: CO5

Objectives: (a) To understand MVC architecture.
(b) To understand and use JSF to replace JSTL and custom tags for a JSP file.
(c) To assess the impact of using a framework instead of pure JSP file for development, maintenance etc. and compare JSF with Spring.
.

Equipment/Instruments: Personal Computer, JDK 1.8 or advance, Netbeans/eclipse, JSF and Hibernetframwork

Theory:

JavaServer Faces
It is a server side component based user interface framework. It is used to develop web applications. It provides a well-defined programming model and consists of rich API and tag libraries. The latest version JSF 2 uses Facelets as its default templating system. It is written in Java.

The JSF API provides components (inputText, commandButtonetc) and helps to manage their states. It also provides server-side validation, data conversion, defining page navigation, provides extensibility, supports for internationalization, accessibility etc.

The JSF Tag libraries are used to add components on the web pages and connect components with objects on the server. It also contains tag handlers that implements the component tag.

With the help of these features and tools, you can easily and effortlessly create server-side user interface.

Benefits of JavaServer Faces
1) It provides clean and clear separation between behavior and presentation of web application. You can write business logic and user interface separately.

2) JavaServer Faces API?s are layered directly on top of the Servlet API. Which enables several various application use cases, such as using different presentation technologies, creating your own custom components directly from the component classes.

3) Including of Facelets technology in JavaServer Faces 2.0, provides massive advantages to it. Facelets is now the preferred presentation technology for building JavaServer Faces based web applications.

Safety and necessary Precautions:
1. Make sure the database server is up and running.
2. Make sure all the hibernet and JSF libraries are put in the classpath
3. Make sure the web server is up and running

4. Handle all necessary compile time Exception

Procedure:
A Simple JavaServer Faces Application
To create a JSF application, we are using NetBeans IDE 8.2. You can also refer to other Java IDEs.

Here, we are creating a project after that we will run to test it's configuration settings. So, let's create a new project fist.

Create a New Project

- Go to file menu and select new Project.-> Select Category Java Web and Project Web Application. -> Enter project name.

- Select Server and Java EE Version.

- Select JSF Framework

- Select Preferred Page Language: Earlier versions of JSF framework are default to JSP for presentation pages. Now, in latest version 2.0 and later JSF has included powerful tool "Facelets". So, here we have selected page language as facelets. We will talk about facelets in more details in next chapter.

- Index.xhtml Page: After finishing, IDE creates a JSF project for you with a default index.xhtml file. Xhtml is a extension of html and used to create facelets page.

- Run: Now, you can run your application by selecting run option after right click on the project. It will produce a default message "Hello from Facelets".

This project includes following files:

- index.xhtml: inside the Web Pages directory
- web.xml: inside the WEB-INF directory

Whenever we run the project, it renders index.xhtml as output. Now, we will create an application which contains two web pages, one bean class and a configuration file.

It requires the following steps in order to develop new application:

- Creating user interface
- Creating managed beans
- Configuring and managing FacesServlet

Observations:

Index.xhtml
```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
      <title>JSF Facelets Hello World</title>
      <style>
          .form-container {
              width: 400px;
```

```
            margin: 0 auto;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
         }
         .form-field {
            margin-bottom: 15px;
         }
         .form-label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
         }
         .form-button {
            padding: 8px 15px;
            background-color: #3498db;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
         }
      </style>
   </h:head>
   <h:body>
      <h1>Welcome to JSF Facelets Demo</h1>

      <div class="form-container">
         <h:form>
            <div class="form-field">
               <h:outputLabel value="Your Name:" for="nameInput" styleClass="form-label" />
               <h:inputText id="nameInput" value="#{userBean.name}" required="true" />
            </div>

            <h:commandButton action="#{userBean.navigate}" value="Submit" styleClass="form-
button" />
         </h:form>
      </div>
   </h:body>
</html>
```

Welcome.xhtml

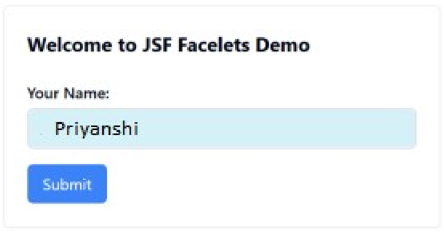
```
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html">
   <h:head>
      <title>Welcome Page</title>
      <style>
         .welcome-container {
            width: 500px;
            margin: 0 auto;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            text-align: center;
         }
         .user-name {
            color: #2980b9;
            font-weight: bold;
         }
         .back-button {
            padding: 8px 15px;
            background-color: #3498db;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            margin-top: 20px;
         }
      </style>
   </h:head>
   <h:body>
      <div class="welcome-container">
         <h1>Hello, <span class="user-name">#{userBean.name}</span>!</h1>
         <p>Welcome to our JSF Facelets application.</p>

         <h:form>
            <h:commandButton action="index" value="Go Back" styleClass="back-button" />
         </h:form>
      </div>
   </h:body>
</html>
```

UserBean.java

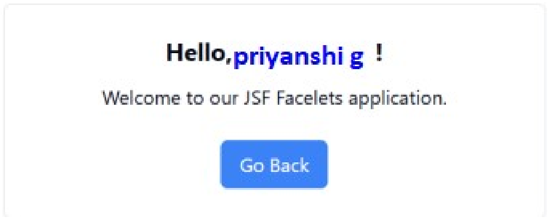
```
package com.example.beans;

import java.io.Serializable;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

@Named
@SessionScoped
public class UserBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;

    public String getName() {
        return name;
    }

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

    public String navigate() {
        // Some business logic could go here
        return "welcome"; // This refers to welcome.xhtml
    }
}
```

Web.xml

```
<web-app version="4.0"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                          http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

    <!-- Application name -->
    <display-name>JSF Facelets Hello World</display-name>

    <!-- Welcome files -->
    <welcome-file-list>
        <welcome-file>index.xhtml</welcome-file>
    </welcome-file-list>

    <!-- JSF configuration -->
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>

    <!-- JSF configuration parameters -->
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>

    <context-param>
        <param-name>javax.faces.FACELETS_SKIP_COMMENTS</param-name>
        <param-value>true</param-value>
    </context-param>
</web-app>
```


Welcome to JSF Facelets Demo

Your Name: Priyanshi


Hello,priyanshi g !

Welcome to our JSF Facelets application.

Suggested Reference:
https://www.oracle.com/java/technologies/javaserverfaces.html

References used by the students:

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition (2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Average (1)	
Marks											