PROGRAM 1: JSP and MVC with Request Dispatcher

**File: index.html**

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<h1>Login Form</h1>

<form method="post" action="MainServ">

Name:<input type="text" name="uname"/><br>

password:<input type="password" name="upass"/><br/>

<input type="submit" value="LOGIN"/>

</form>

</body>

</html>

**File:MainServ.java**

package servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.RequestDispatcher;

/\*\*

\*

\* @author Deepthi

\*/

public class MainServ extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException

{

PrintWriter out=response.getWriter();

String n=request.getParameter("uname");

String p=request.getParameter("upass");

if(n.equals("admin")&&p.equals("root"))

{

RequestDispatcher rd=request.getRequestDispatcher("SecondServ");

rd.forward(request, response);

}

else

{

out.print("sorry UserName or Passwod Error!");

RequestDispatcher rd=request.getRequestDispatcher("/index.html");

}

}

}

**File: SecondServ.java**

package servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class SecondServ extends HttpServlet

{

@Override

protected void doPost(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException

{

PrintWriter out=response.getWriter();

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

out.println("<h1>Welcome!!!!</h1>"+name);

}

}

**File: web.xml**

<?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" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd">

<servlet>

<servlet-name>MainServ</servlet-name>

<servlet-class>servlet.MainServ</servlet-class>

</servlet>

<servlet>

<servlet-name>SecondServ</servlet-name>

<servlet-class>servlet.SecondServ</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MainServ</servlet-name>

<url-pattern>/MainServ</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>SecondServ</servlet-name>

<url-pattern>/SecondServ</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>

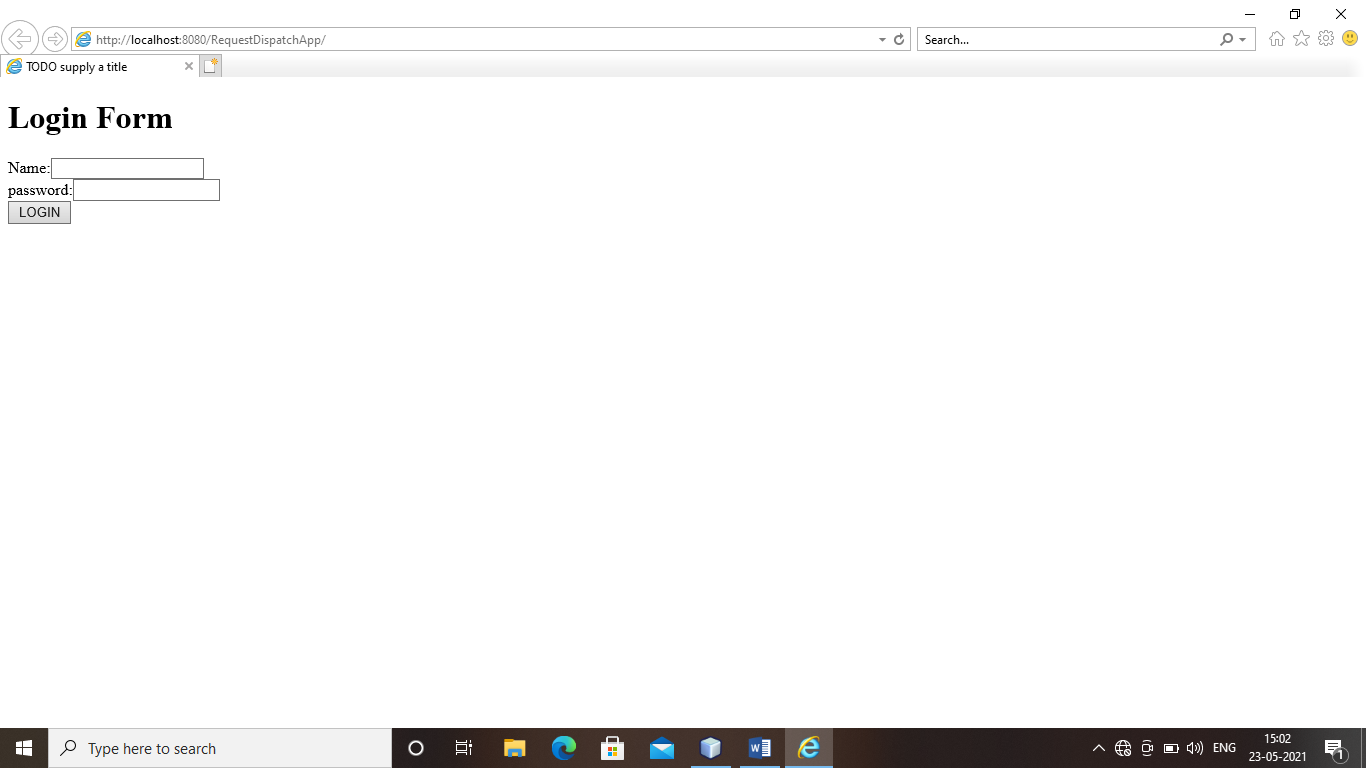
30

</session-timeout>

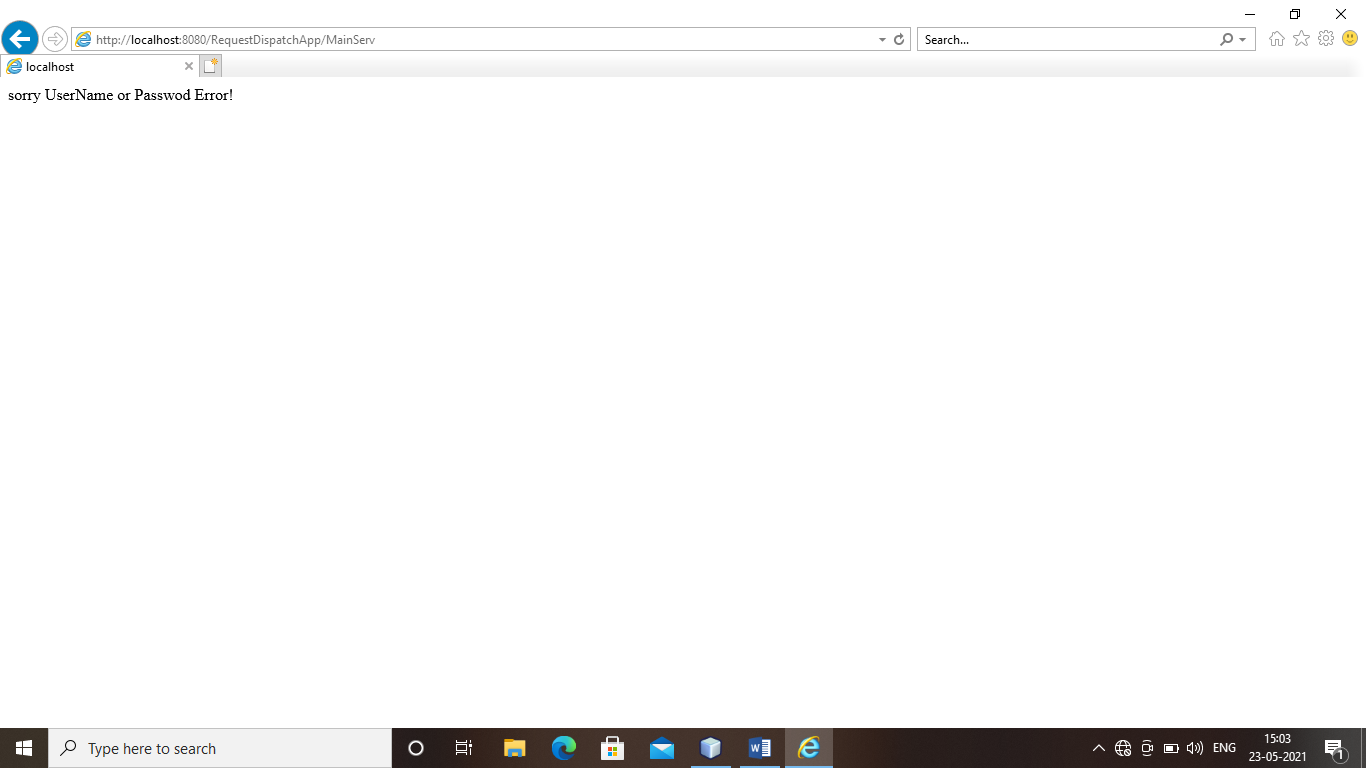
</session-config>

</web-app>

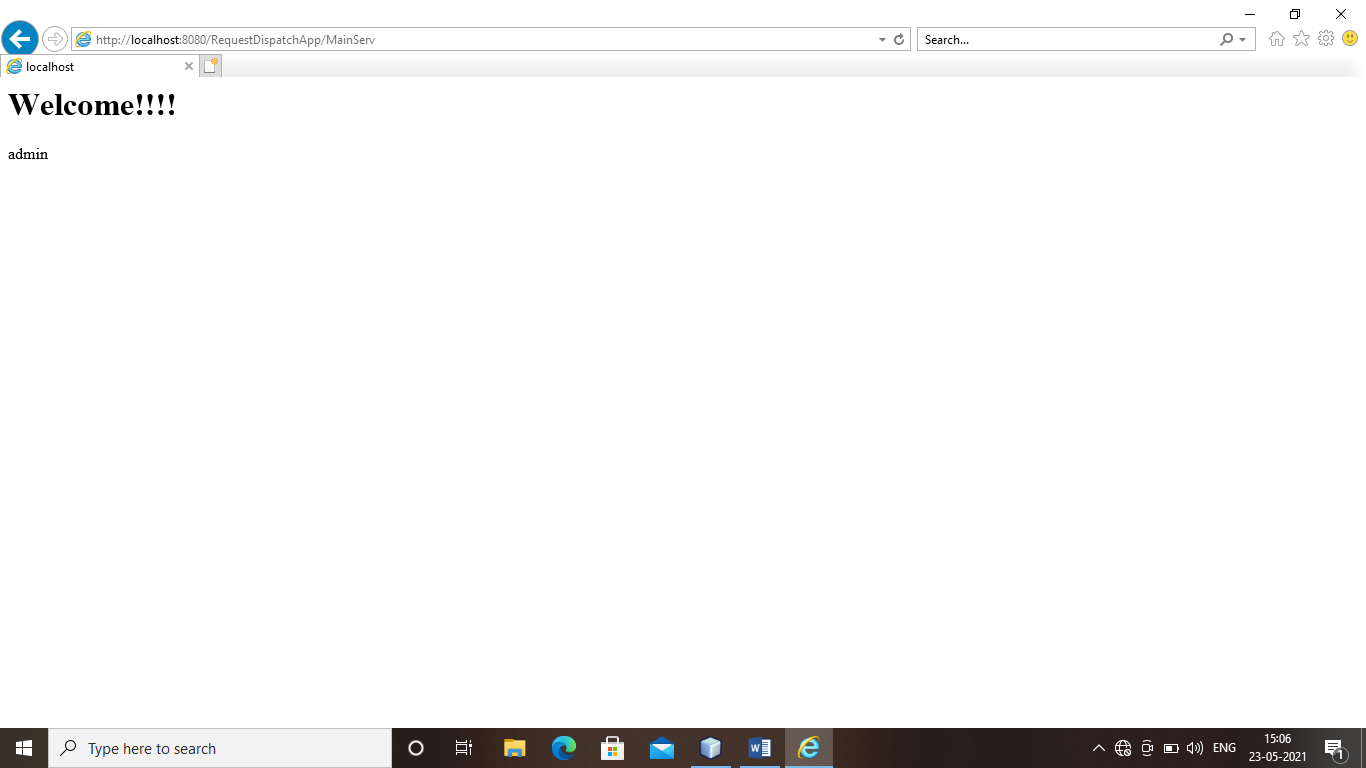
OUTPUT-1:



OUTPUT-2:



OUTPUT-3:



PROGRAM 2: JSF AND JSP PAGES USING ALL HTML AND CORE RENDER KIT

Index.xhtml

<?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>User Form</title>

</h:head>

<h:body>

<h:form>

<h:outputLabel for="username">User Name</h:outputLabel>

<h:inputText id="username" value="#{user.name}" required="true" requiredMessage="User Name is required" /><br/>

<h:commandButton id="submit-button" value="Submit" action="response.xhtml"/>

</h:form>

</h:body>

</html>

User.java

import javax.faces.bean.ManagedBean;

import javax.faces.bean.RequestScoped;

@ManagedBean

@RequestScoped

public class User {

String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

Response.xhtml

<?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>Welcome Page</title>

</h:head>

<h:body>

<h2>Hello, <h:outputText value="#{user.name}"></h:outputText></h2>

</h:body>

</html>

WEB.xml

<?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" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd">

<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>

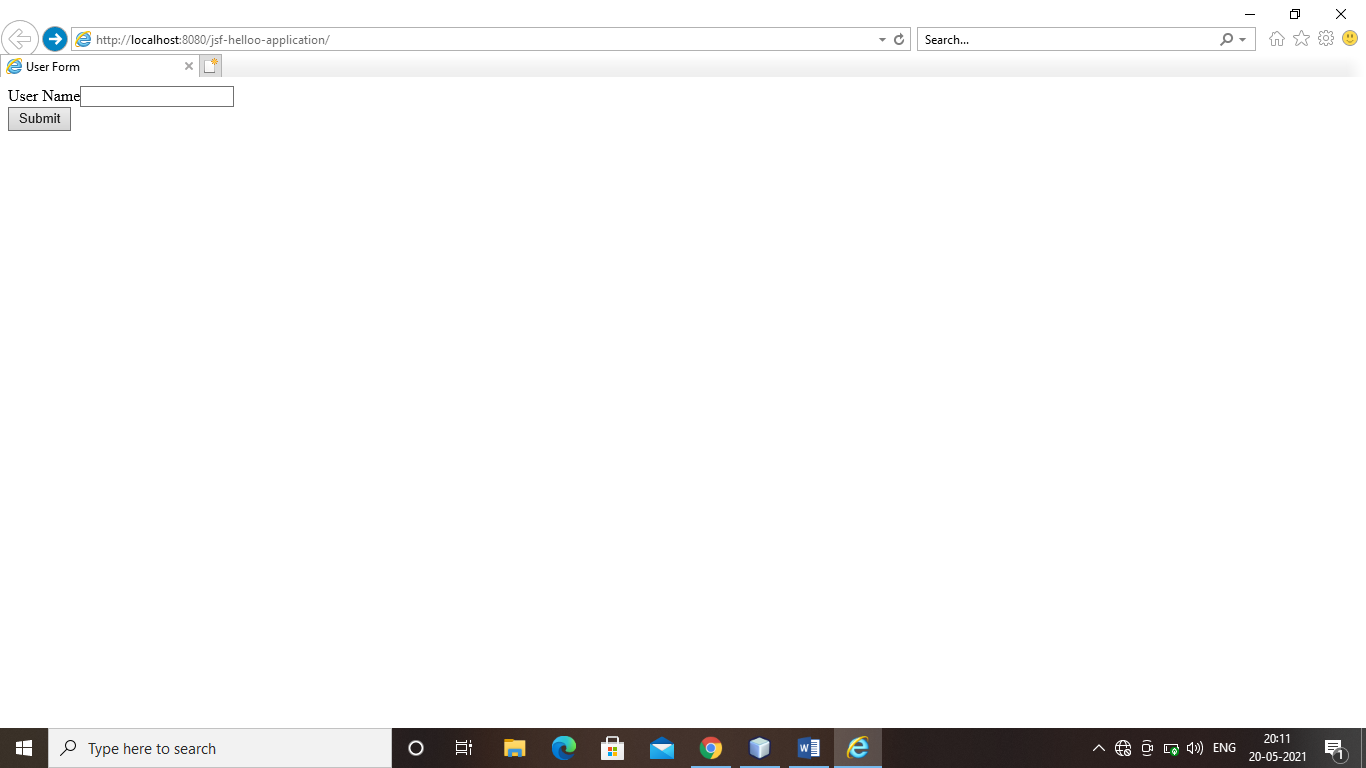
<welcome-file-list>

<welcome-file>faces/index.xhtml</welcome-file>

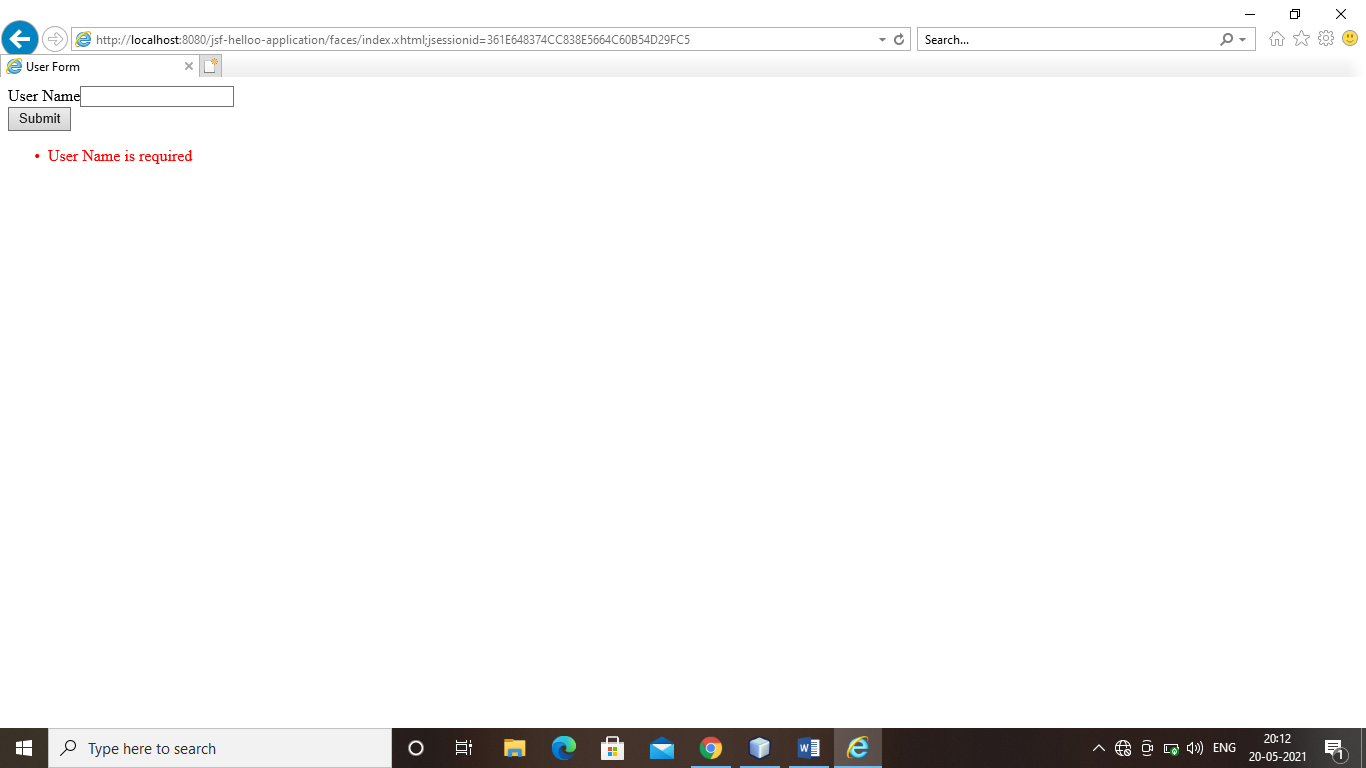
</welcome-file-list>

</web-app>

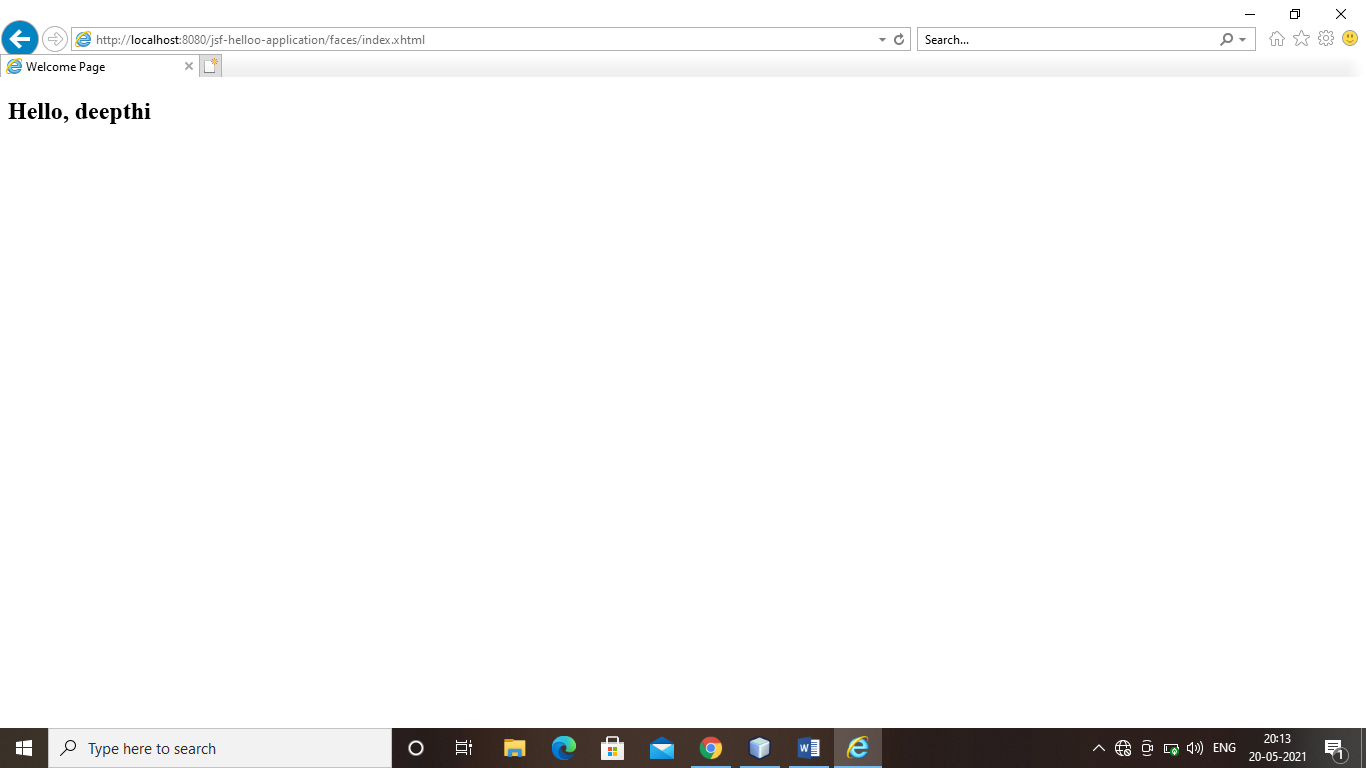
OUTPUT-1:



OUTPUT-2:



OUTPUT-3:



PROGRAM 3: ACTION AND FORMS

**Person.java**

**public class Person {**

**private String firstName;**

**private String lastName;**

**private String email;**

**private int age;**

**public String getFirstName() {**

**return firstName;**

**}**

**public void setFirstName(String firstName) {**

**this.firstName = firstName;**

**}**

**public String getLastName() {**

**return lastName;**

**}**

**public void setLastName(String lastName) {**

**this.lastName = lastName;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public int getAge() {**

**return age;**

**}**

**public void setAge(int age) {**

**this.age = age;**

**}**

**public String toString() {**

**return "First Name: " + getFirstName() + " Last Name: " + getLastName() +**

**" Email: " + getEmail() + " Age: " + getAge() ;**

**}**

**}**

**register.jsp**

**<%@ taglib prefix="s" uri="/struts-tags" %>**

**<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Register</title>**

**</head>**

**<body>**

**<h3>Register for a prize by completing this form.</h3>**

**<s:form action="register">**

**<s:textfield name="personBean.firstName" label="First name" />**

**<s:textfield name="personBean.lastName" label="Last name" />**

**<s:textfield name="personBean.email" label ="Email"/>**

**<s:textfield name="personBean.age" label="Age" />**

**<s:submit/>**

**</s:form>**

**</body>**

**</html>**

**Register.java Struts 2 Action Class**

package org.apache.struts.register.action;

import com.opensymphony.xwork2.ActionSupport;

import org.apache.struts.register.model.Person;

public class Register extends ActionSupport {

private static final long serialVersionUID = 1L;

private Person personBean;

public String execute() throws Exception {

//call Service class to store personBean's state in database

return SUCCESS;

}

public Person getPersonBean() {

return personBean;

}

public void setPersonBean(Person person) {

personBean = person;

}

}

**thankyou.jsp**

**<%@ taglib prefix="s" uri="/struts-tags" %>**

**<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Registration Successful</title>**

**</head>**

**<body>**

**<h3>Thank you for registering for a prize.</h3>**

**<p>Your registration information: <s:property value="personBean" /> </p>**

**<p><a href="<s:url action='index' />" >Return to home page</a>.</p>**

**</body>**

**</html>**

**action node for struts.xml**

<action name="register" class="org.apache.struts.register.action.Register" method="execute">

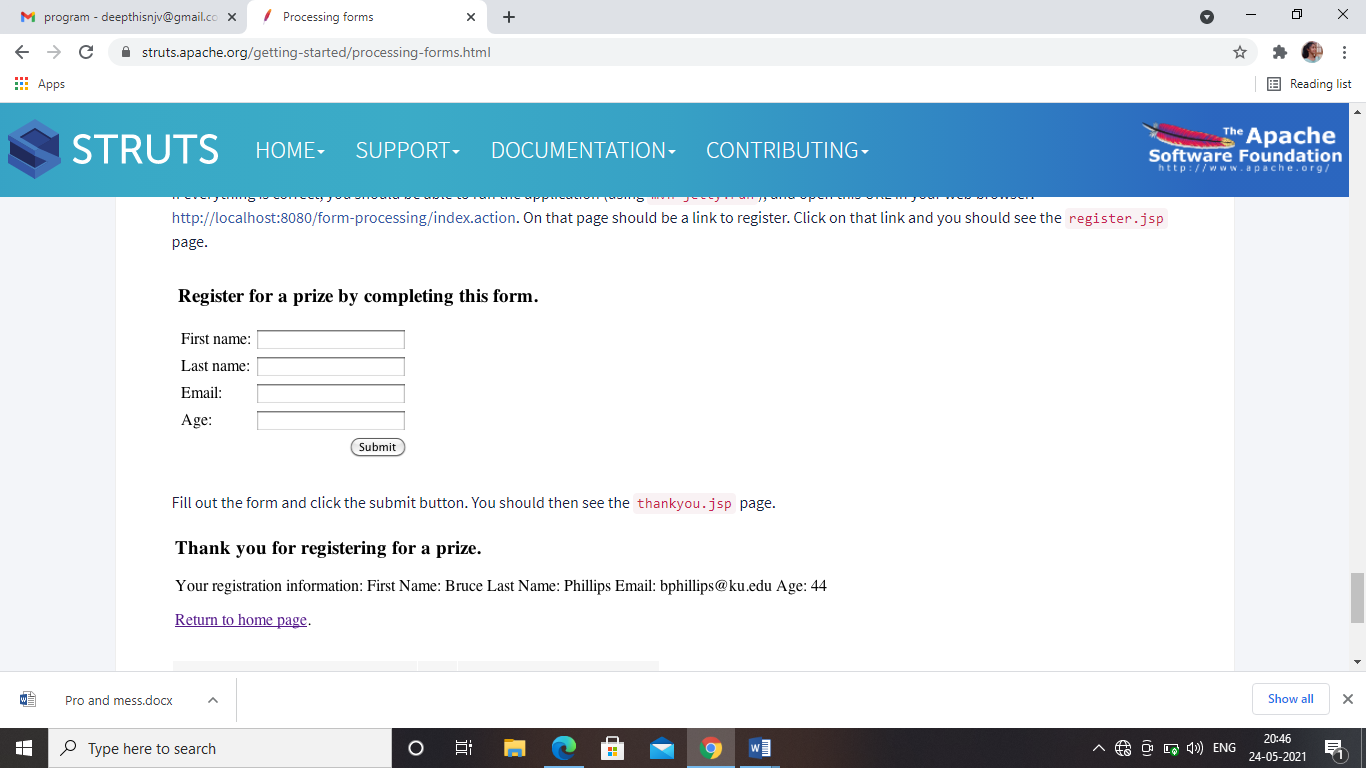
<result name="success">/thankyou.jsp</result>

</action>

**Link to register.jsp**

**<p><a href="register.jsp">Please register</a> for our prize drawing.</p>**

OUTPUT:



PROGRAM 4: PROPERTIES AND MESSAGES

**Property.java**

import java.util.\*;

public class Property {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Properties capitals = new Properties();

Set states;

String str;

capitals.put("Illinois", "Springfield");

capitals.put("Missouri", "Jefferson City");

capitals.put("Washington", "Olympia");

capitals.put("California", "Sacramento");

capitals.put("Indiana", "Indianapolis");

// Show all states and capitals in hashtable.

states = capitals.keySet(); // get set-view of keys

Iterator itr = states.iterator();

while(itr.hasNext()) {

str = (String) itr.next();

System.out.println("The capital of " + str + " is " +

capitals.getProperty(str) + ".");

}

System.out.println();

// look for state not in list -- specify default

str = capitals.getProperty("Florida", "Not Found");

System.out.println("The capital of Florida is " + str + ".");

}

}

**OUTPUT:**

The capital of Missouri is Jefferson City.

The capital of Illinois is Springfield.

The capital of Indiana is Indianapolis.

The capital of California is Sacramento.

The capital of Washington is Olympia.

The capital of Florida is Not Found.

BUILD SUCCESSFUL (total time: 0 seconds)

**Messages**

import com.tutorialspoint.entity.Book;

import com.tutorialspoint.stateless.LibraryPersistentBeanRemote;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.List;

import java.util.Properties;

import javax.jms.ObjectMessage;

import javax.jms.Queue;

import javax.jms.QueueConnection;

import javax.jms.QueueConnectionFactory;

import javax.jms.QueueSender;

import javax.jms.QueueSession;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class EJBTester {

BufferedReader brConsoleReader = null;

Properties props;

InitialContext ctx;

{

props = new Properties();

try {

props.load(new FileInputStream("jndi.properties"));

} catch (IOException ex) {

ex.printStackTrace();

}

try {

ctx = new InitialContext(props);

} catch (NamingException ex) {

ex.printStackTrace();

}

brConsoleReader =

new BufferedReader(new InputStreamReader(System.in));

}

public static void main(String[] args) {

EJBTester ejbTester = new EJBTester();

ejbTester.testMessageBeanEjb();

}

private void showGUI() {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Welcome to Book Store");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("Options \n1. Add Book\n2. Exit \nEnter Choice: ");

}

private void testMessageBeanEjb() {

try {

int choice = 1;

Queue queue = (Queue) ctx.lookup("/queue/BookQueue");

QueueConnectionFactory factory =

(QueueConnectionFactory) ctx.lookup("ConnectionFactory");

QueueConnection connection = factory.createQueueConnection();

QueueSession session =

connection.createQueueSession(false, QueueSession.AUTO\_ACKNOWLEDGE);

QueueSender sender = session.createSender(queue);

while (choice != 2) {

String bookName;

showGUI();

String strChoice = brConsoleReader.readLine();

choice = Integer.parseInt(strChoice);

if (choice == 1) {

System.out.print("Enter book name: ");

bookName = brConsoleReader.readLine();

Book book = new Book();

book.setName(bookName);

ObjectMessage objectMessage =

session.createObjectMessage(book);

sender.send(objectMessage);

} else if (choice == 2) {

break;

}

}

LibraryPersistentBeanRemote libraryBean =

(LibraryPersistentBeanRemote)

ctx.lookup("LibraryPersistentBean/remote");

List<Book> booksList = libraryBean.getBooks();

System.out.println("Book(s) entered so far: " + booksList.size());

int i = 0;

for (Book book:booksList) {

System.out.println((i+1)+". " + book.getName());

i++;

}

} catch (Exception e) {

System.out.println(e.getMessage());

e.printStackTrace();

}finally {

try {

if(brConsoleReader !=null) {

brConsoleReader.close();

}

} catch (IOException ex) {

System.out.println(ex.getMessage());

}

}

} }

**OUTPUT:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to Book Store

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Options

1. Add Book

2. Exit

Enter Choice: 1

Enter book name: Learn EJB

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to Book Store

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Options

1. Add Book

2. Exit

Enter Choice: 2

Book(s) entered so far: 2

1. learn java

1. learn EJB

BUILD SUCCESSFUL (total time: 15 seconds)

PROGRAM 5: CREATING WEB CLIENT AND SESSION BEAN

WEB CLIENT

Addclient.java

package addclient;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

/\*\*

\*

\* @author Deepthi

\*/

@WebService(serviceName = "addclient")

public class addclient {

/\*\*

\* Web service operation

\*/

@WebMethod(operationName = "add")

public int add(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {

//TODO write your implementation code here:

int c=a+b;

return c;

}

}

index.jsp

<html>

<head>

<title>JSP PAGE</title>

</head>

<body>

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

Enter First Number:<input type="text" name="fst"/><br/>

Enter Second Number:<input type="text" name="snd"/><br/>

<input type="submit" value="Add"/>

</form>

</body>

</html>

action.jsp

<%--

Document : action

Created on : 27 May, 2021, 11:16:21 AM

Author : Deepthi

--%>

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

<!DOCTYPE html>

<html>

<head>

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

<title>JSP Page</title>

</head>

<body>

<h1>Hello World!</h1>

<%-- start web service invocation --%><hr/>

<%

String a1=request.getParameter("fst");

String b1=request.getParameter("snd");

int aa=Integer.parseInt(a1);

int bb=Integer.parseInt(b1);

try {

addclient.Addclient\_Service service = new addclient.Addclient\_Service();

addclient.Addclient port = service.getAddclientPort();

// TODO initialize WS operation arguments here

int a = aa;

int b = bb;

// TODO process result here

int result = port.add(a, b);

out.println("Result = "+result);

} catch (Exception ex) {

// TODO handle custom exceptions here

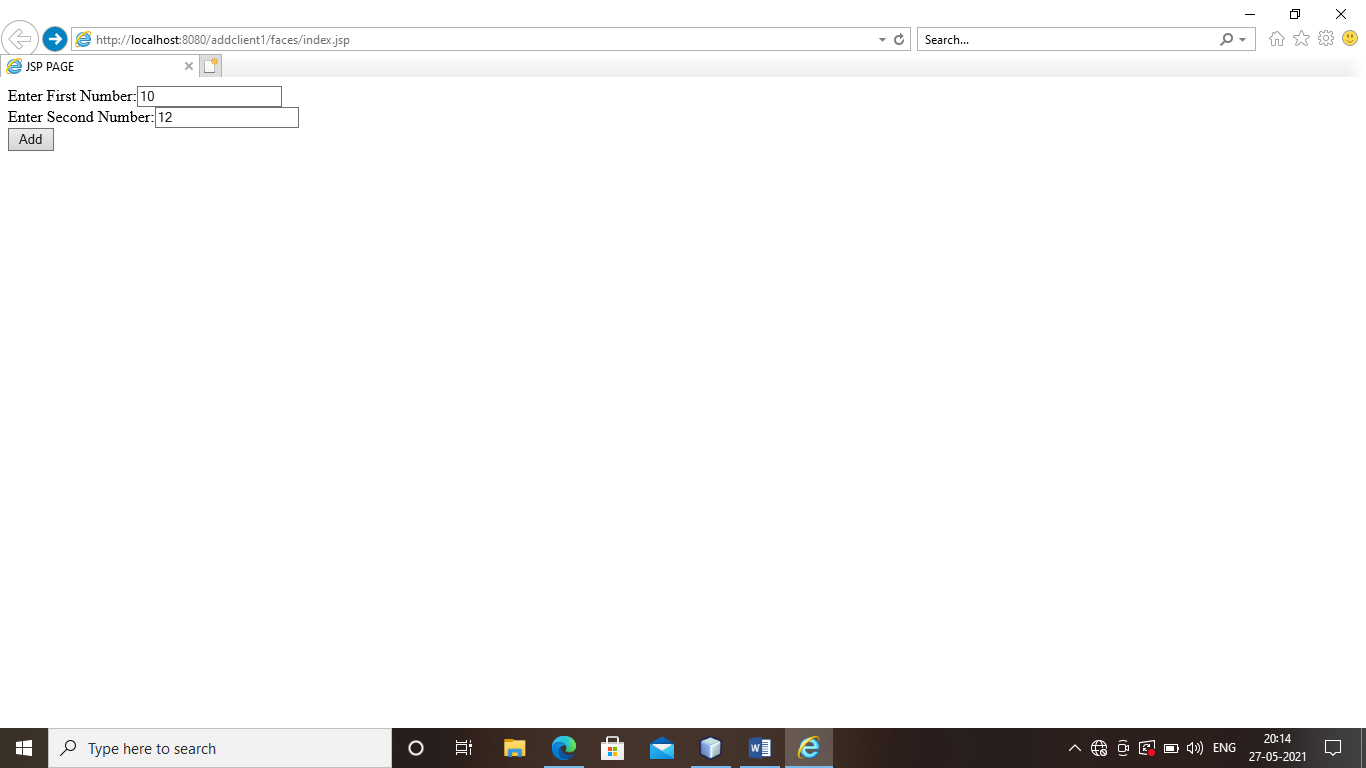
}

%>

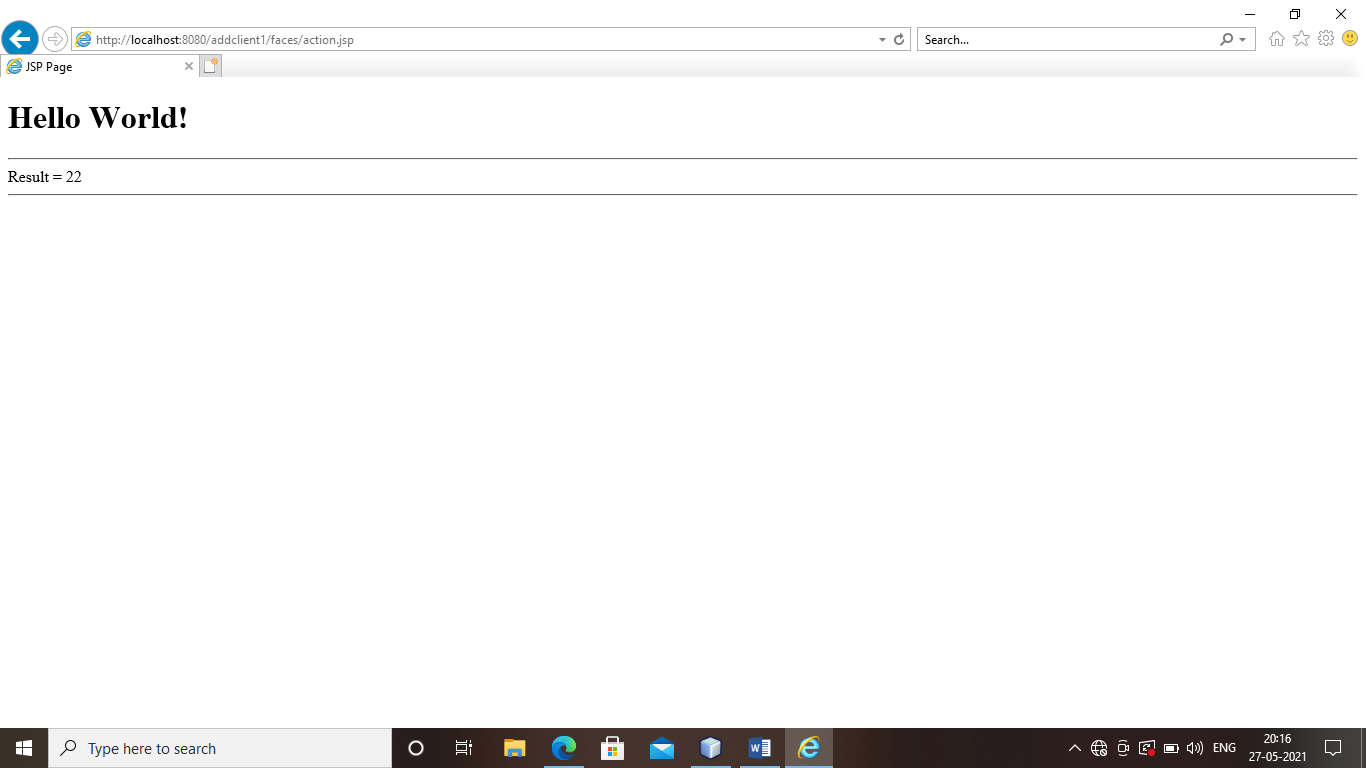
<%-- end web service invocation --%><hr/>

</body>

</html>

OUTPUT 1

OUTPUT 2:



SESSION BEAN

Index.jsp

<!DOCTYPE html>

<!--

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

-->

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<h1> Bank Transaction Request Form</h1>

<hr><br>

<form action="result" method="POST" >

<table align="Center">

<tr><td></td></tr>

<tr><td>Enter the amount in rupees:

<input type="text" name="amt" size="10" required></td></tr>

<br>

<tr><td><b>Select Your Choice:</b></tr></td>

<tr><td><input type="radio" name="group1" value="dep">Deposite</tr></td>

<tr><td><input type="radio" name="group1" value="with">Withdraw<br></tr></td>

<tr><td>

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

<input type="reset" value="Reset">

<tr><td></tr></td>

</form>

</table>

</body>

</html>

NewSessionBean.java

package YJ;

import javax.ejb.Stateless;

/\*\*

\*

\* @author Deepthi

\*/

@Stateless

public class NewSessionBean implements NewSessionBeanLocal {

float bal=0;

@Override

public Float deposit(float amt) {

bal+=amt;

return bal;

}

// Add business logic below. (Right-click in editor and choose

// "Insert Code > Add Business Method")

@Override

public Float withdraw(float amt) {

bal-=amt;

return bal;

}

}

Result.java

package YJ;

import java.io.IOException;

import java.io.PrintWriter;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author Deepthi

\*/

@WebServlet(name="result",urlPatterns={"/result"})

public class result extends HttpServlet {

@EJB

private NewSessionBeanLocal newSessionBean;

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

try (PrintWriter out = response.getWriter()) {

float bal=0;

String s1=request.getParameter("amt");

String s2=request.getParameter("group1");

/\* TODO output your page here. You may use following sample code. \*/

out.println("<!DOCTYPE html>");

out.println("<html>");

if(s1 !=null) {

Float amt=new Float(s1);

if(s2.equals("dep"))

bal=newSessionBean.deposit(amt.floatValue());

else if(s2.equals("with"))

bal=newSessionBean.withdraw(amt.floatValue());

else

out.println("<h3>Please select your choice</h3><br>");

}

out.println("<head>");

out.println("<title>Servlet result</title>");

out.println("</head>");

out.println("<body>");

out.println("<h1>Your Current Balance is: "+bal+"</h1><br>");

out.println("<a href=index.html>Back</a>");

out.println("</body>");

out.println("</html>");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

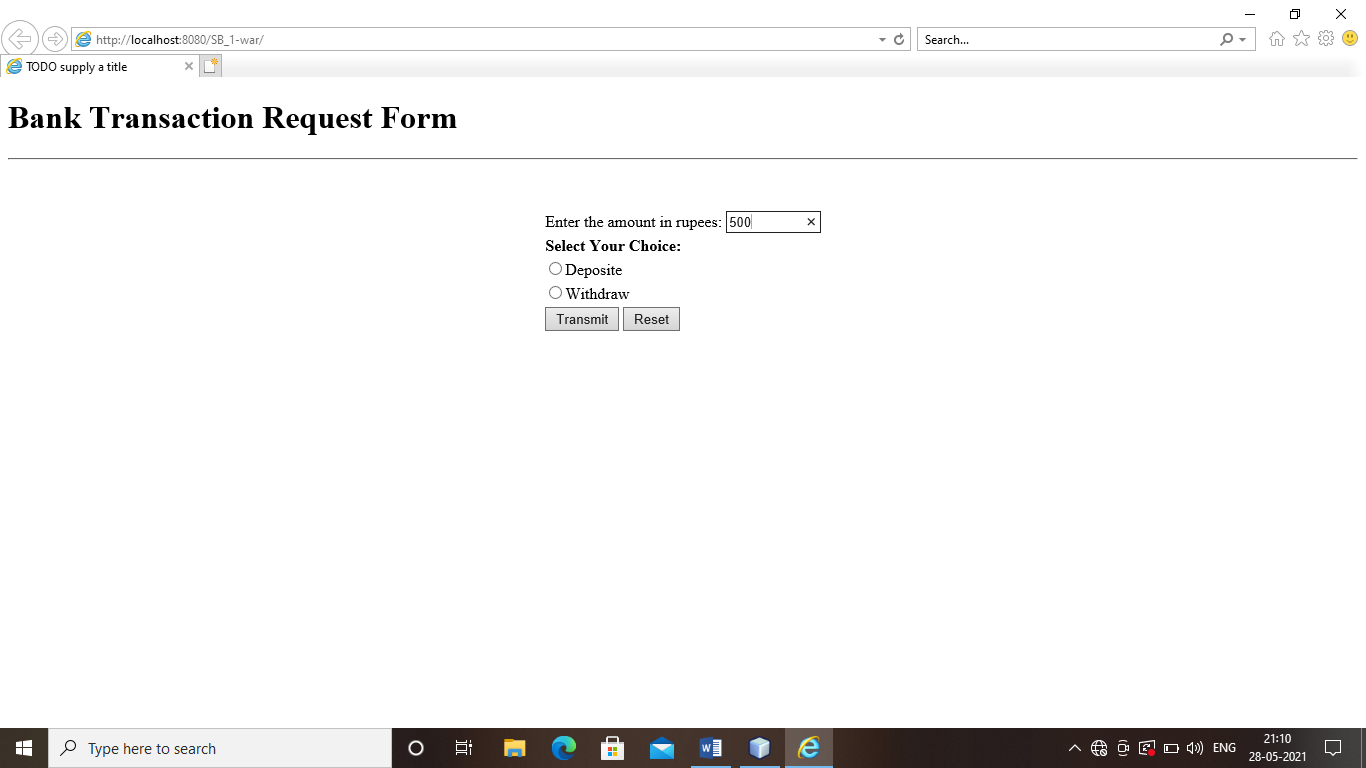
public String getServletInfo() {

return "Short description";

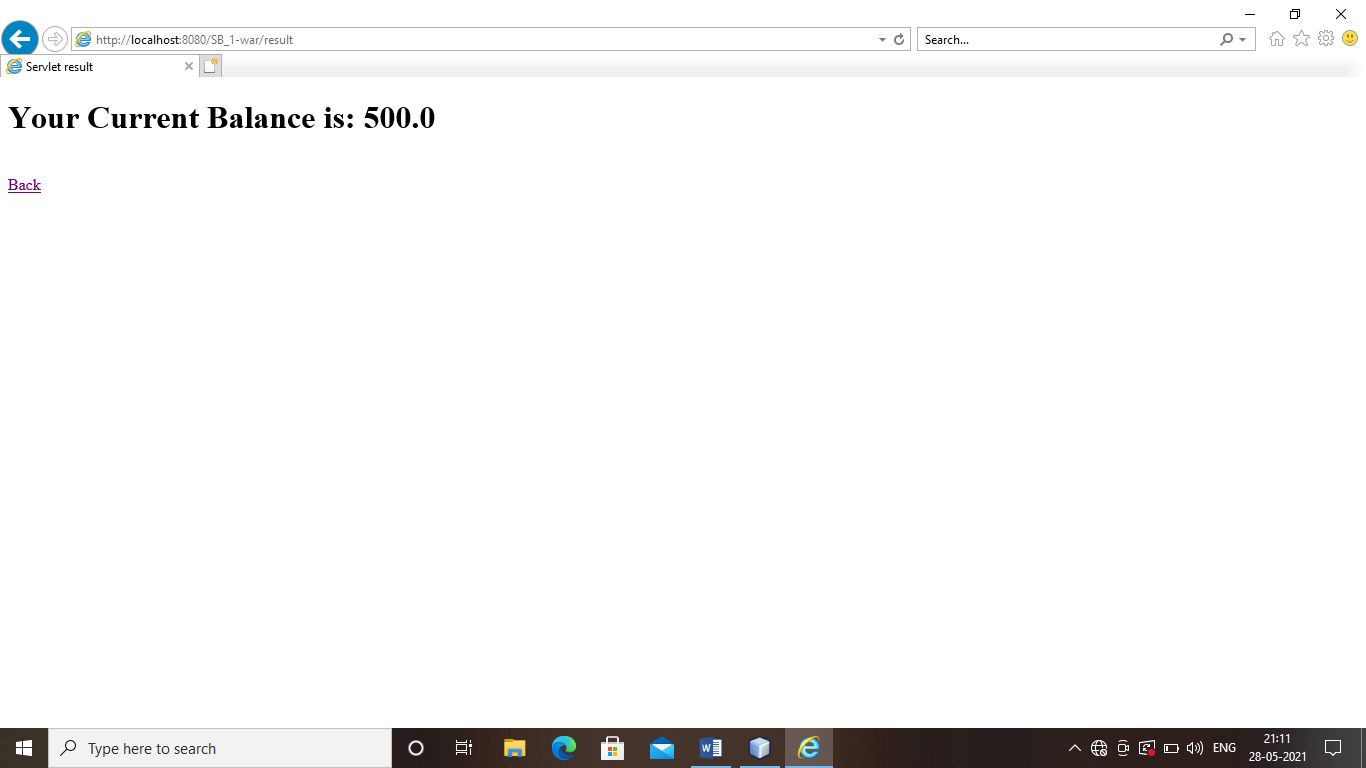
}// </editor-fold>

}

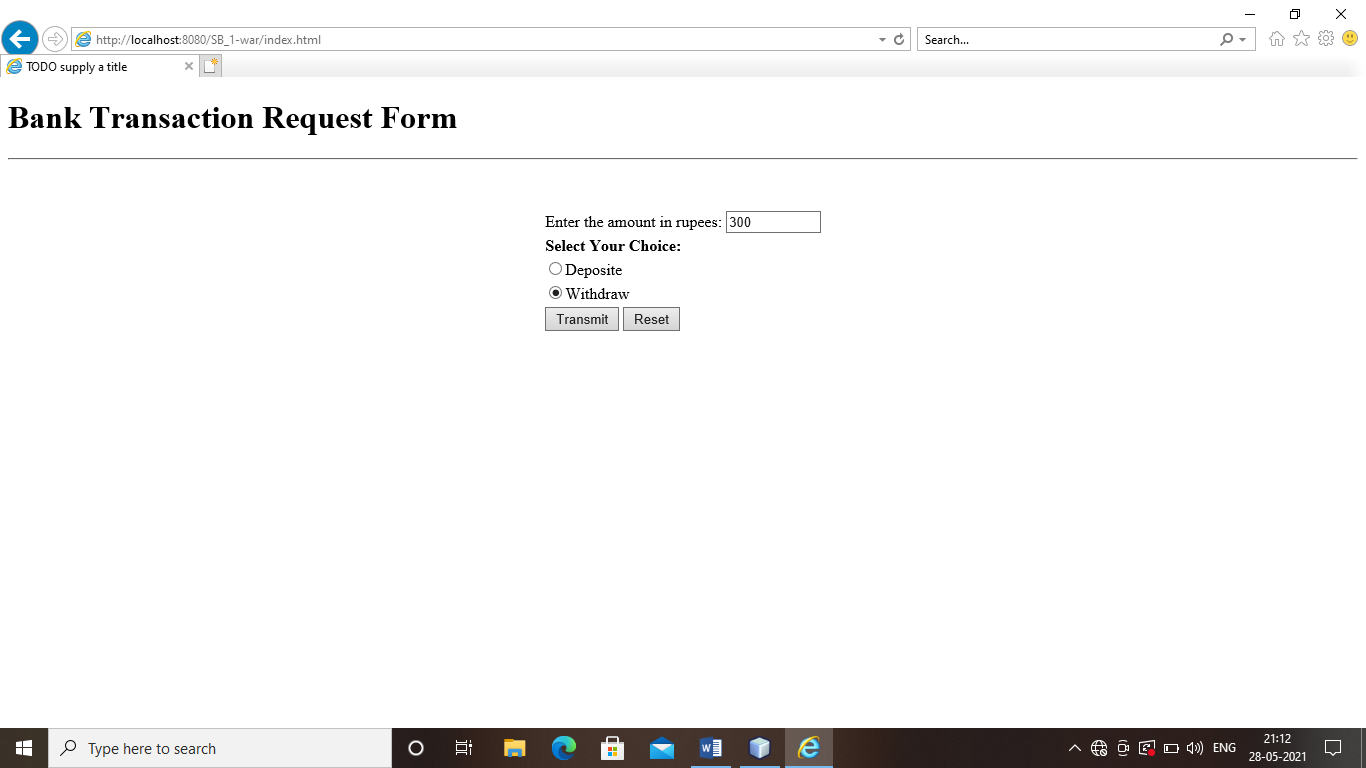
**OUTPUT 1:**



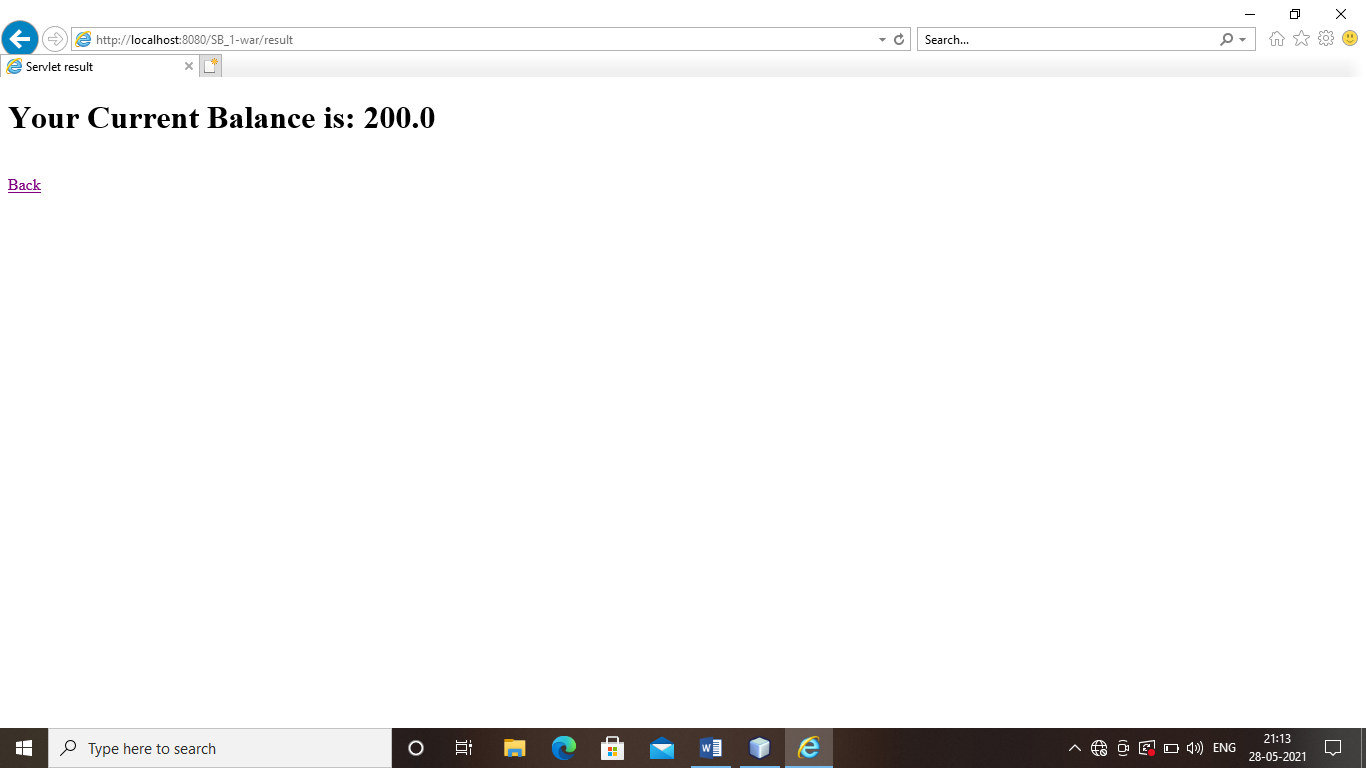
**OUTPUT 2:**



**OUTPUT 3:**



**OUTPUT 4:**



PROGRAM 6: BEAN MANAGED TRANSACTION AND CONTAINER MANAGED TRANSACTION

BEAN MANAGED TRANSACTION

**Employee**

package com.javacodegeeks.snippets.enterprise;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Employee implements Serializable {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private Long id;

private String name;

private String surname;

private String title;

private Double salary;

private Date created;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public Date getCreated() {

return created;

}

public void setCreated(Date created) {

this.created = created;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", surname=" + surname

+ ", title=" + title + ", salary="+salary+ ", created=" + created+"]";

}

}

**EmployeeService**

package com.javacodegeeks.snippets.enterprise;

import java.util.Collection;

import java.util.Date;

import javax.annotation.Resource;

import javax.ejb.EJBException;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.Query;

import javax.transaction.UserTransaction;

@Stateless

@TransactionManagement(TransactionManagementType.BEAN)

public class EmployeeService implements EmployeeServiceLocal, EmployeeServiceRemote {

@PersistenceContext(unitName = "TutorialPU")

EntityManager entityManager;

@Resource

UserTransaction tx;

public EmployeeService() {

}

public Employee createEmployee(String name, String surname, String title, double salary) {

Employee employee = new Employee();

employee.setName(name);

employee.setSurname(surname);

employee.setTitle(title);

employee.setSalary(salary);

employee.setCreated(new Date());

try {

try {

tx.begin();

entityManager.persist(employee);

} finally {

tx.commit();

}

} catch (Exception e) {

// handle all the tx.begin()/commit() exceptions

throw new EJBException(e);

}

return employee;

}

public void removeEmployee(long id) {

Employee employee = findEmployee(id);

if (employee != null) {

try {

try {

tx.begin();

entityManager.remove(employee);

} finally {

tx.commit();

}

} catch (Exception e) {

// handle all the tx.begin()/commit() exceptions

throw new EJBException(e);

}

}

}

public Employee promoteEmployee(long id, String newTitle, double newSalary) {

Employee employee = entityManager.find(Employee.class, id);

if (employee != null) {

employee.setTitle(newTitle);

employee.setSalary(newSalary);

try {

try {

tx.begin();

entityManager.merge(employee);

} finally {

tx.commit();

}

} catch (Exception e) {

// handle all the tx.begin()/commit() exceptions

throw new EJBException(e);

}

}

return employee;

}

public Employee findEmployee(long id) {

return entityManager.find(Employee.class, id);

}

public Collection<Employee> findAllEmployees() {

Query query = entityManager.createQuery("SELECT e FROM Employee e");

return (Collection<Employee>) query.getResultList();

}

}

**EmployeeServiceLocal**

package com.javacodegeeks.snippets.enterprise;

import java.util.Collection;

import javax.ejb.Local;

@Local

public interface EmployeeServiceLocal {

public Employee createEmployee(String name, String surname, String title, double salary);

public void removeEmployee(long id);

public Employee promoteEmployee(long id, String newTitle, double newSalary);

public Employee findEmployee(long id);

public Collection<Employee> findAllEmployees();

}

**EmployeeServiceRemot**e

package com.javacodegeeks.snippets.enterprise;

import java.util.Collection;

import javax.ejb.Remote;

@Remote

public interface EmployeeServiceRemote {

public Employee createEmployee(String name, String surname, String title, double salary);

public void removeEmployee(long id);

public Employee promoteEmployee(long id, String newTitle, double newSalary);

public Employee findEmployee(long id);

public Collection<Employee> findAllEmployees();

}

**persistence.xml**

<persistence xmlns="http://java.sun.com/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence\_1\_0.xsd" version="1.0">

<persistence-unit name="TutorialPU" >

<jta-data-source>java:/DefaultDS</jta-data-source>

<properties>

<property name="hibernate.hbm2ddl.auto" value="create-drop"/>

</properties>

<!--

<properties>

<property name="hibernate.dialect" value="org.hibernate.dialect.HSQLDialect"/>

<property name="hibernate.hbm2ddl.auto" value="update"/>

<property name="hibernate.connection.driver\_class" value="org.hsqldb.jdbcDriver"/>

<property name="hibernate.connection.username" value="sa"/>

<property name="hibernate.connection.password" value=""/>

<property name="hibernate.connection.url" value="jdbc:hsqldb:data/tutorial"/>

</properties>

-->

</persistence-unit>

</persistence>

**application.xml**

<?xml version="1.0" encoding="UTF-8"?>

<application xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/application\_1\_4.xsd"

version="1.4">

<display-name>ExampleEJB3</display-name>

<module>

<java>exampleEJB3-persistence.jar</java>

</module>

<module>

<ejb>exampleEJB3.jar</ejb>

</module>

</application>

**OUTPUT:**

[id=1, name=Byron, surname=Kiourtzoglou, title=Master Software Engineer, salary=2000.0, created=2011-12-03 17:31:30.203]

Employee [id=1, name=Byron, surname=Kiourtzoglou, title=Principal Software Engineer, salary=3000.0, created=2011-12-03 17:31:30.203]

CONTAINER MANAGED TRANSACTION

**CartBean.java**

package com.javacodegeeks.example.beans;

import java.util.ArrayList;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.ejb.AfterBegin;

import javax.ejb.AfterCompletion;

import javax.ejb.BeforeCompletion;

import javax.ejb.Remove;

import javax.ejb.Stateful;

import javax.ejb.TransactionAttribute;

import javax.ejb.TransactionAttributeType;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

/\*\*

\*

\* @author jGauravGupta

\*/

@Stateful

@TransactionManagement(value=TransactionManagementType.CONTAINER)

public class CartBean {

private ArrayList items;

@PostConstruct

public void init() {

items = new ArrayList();

System.out.println("CartBean: init");

}

@PreDestroy

public void destroy() {

System.out.println("CartBean: destroy");

}

@Remove

public void checkOut() {

// Release any resources.

System.out.println("Cart checkout...");

}

public void addItem(String item) {

getItems().add(item);

System.out.println(item + " item added to cart");

}

public void removeItem(String item) {

getItems().remove(item);

System.out.println(item + " item removed from cart");

}

public ArrayList getItems() {

return items;

}

@AfterBegin

private void afterBegin(){

System.out.println("A new transaction has started.");

}

@BeforeCompletion

private void beforeCompletion(){

System.out.println("A transaction is about to be committed.");

}

@AfterCompletion

private void afterCompletion(boolean committed) {

System.out.println("a transaction commit protocol has completed, and tells the instance whether the transaction has been committed or rolled back , based on committed value : " + committed);

}

}

**NO\_TX\_Client\_Tester.java**

package com.javacodegeeks.example.tester.non\_tx;

import com.javacodegeeks.example.beans.CartBean;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author jGauravGupta

\*/

@WebServlet(name = "NO\_TX\_Client\_Tester", urlPatterns = {"/NO\_TX\_Client\_Tester"})

public class NO\_TX\_Client\_Tester extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

try (PrintWriter out = response.getWriter()) {

CartBean cartBean = lookupCartBeanBean();

cartBean.addItem("Smart Watch");

cartBean.addItem("iPhone");

cartBean.addItem("Shoes");

out.println("Cart Item Size : " + cartBean.getItems().size());

cartBean.checkOut();

}

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

private CartBean lookupCartBeanBean() {

try {

Context c = new InitialContext();

return (CartBean) c.lookup("java:global/CMT\_Example/CartBean!com.javacodegeeks.example.beans.CartBean");

} catch (NamingException ne) {

Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);

throw new RuntimeException(ne);

}

}

}

**OUTPUT:**

Info: A new transaction has started.

Info: Smart Watch item added to cart

Info: A transaction is about to be committed.

Info: a transaction commit protocol has completed, and tells the instance whether the transaction has been committed or rolled back , based on committed value : true

Info: A new transaction has started.

Info: iPhone item added to cart

Info: A transaction is about to be committed.

Info: a transaction commit protocol has completed, and tells the instance whether the transaction has been committed or rolled back , based on committed value : true

Info: A new transaction has started.

Info: Shoes item added to cart

Info: A transaction is about to be committed.

Info: a transaction commit protocol has completed, and tells the instance whether the transaction has been committed or rolled back , based on committed value : true

Info: A new transaction has started.

Info: Cart checkout...

**TX\_Client\_Tester.java**

package com.javacodegeeks.example.tester.tx;

import com.javacodegeeks.example.beans.CartBean;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.annotation.Resource;

import javax.ejb.Singleton;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.transaction.HeuristicMixedException;

import javax.transaction.HeuristicRollbackException;

import javax.transaction.NotSupportedException;

import javax.transaction.RollbackException;

import javax.transaction.SystemException;

import javax.transaction.UserTransaction;

/\*\*

\*

\* @author jGauravGupta

\*/

@Singleton

@TransactionManagement(TransactionManagementType.BEAN)

public class CartProcess {

@Resource

private UserTransaction ut;

public void executeCartProcess() {

try {

Context c = new InitialContext();

CartBean cartBean = (CartBean) c.lookup("java:global/CMT\_Example/CartBean!com.javacodegeeks.example.beans.CartBean");

ut.begin();

cartBean.addItem("Smart Watch");

cartBean.addItem("iPhone");

cartBean.addItem("Shoes");

System.out.println("Cart Item Size : " + cartBean.getItems().size());

ut.commit();

cartBean.checkOut();

} catch (NamingException ex) {

Logger.getLogger(CartProcess.class.getName()).log(Level.SEVERE, null, ex);

} catch (RollbackException | HeuristicMixedException | HeuristicRollbackException | SecurityException | IllegalStateException | SystemException | NotSupportedException ex) {

try {

ut.rollback();

Logger.getLogger(CartProcess.class.getName()).log(Level.SEVERE, null, ex);

} catch (IllegalStateException | SecurityException | SystemException ex1) {

Logger.getLogger(CartProcess.class.getName()).log(Level.SEVERE, null, ex1);

}

}

}

}

**OUTPUT:**

Info: CartBean: init

Info: A new transaction has started.

Info: Smart Watch item added to cart

Info: iPhone item added to cart

Info: Shoes item added to cart

Info: Cart Item Size : 3

Info: A transaction is about to be committed.

Info: a transaction commit protocol has completed, and tells the instance whether the transaction has been committed or rolled back , based on committed value : true

Info: A new transaction has started.

PROGRAM 7: OBJECT RELATION MAPPING AND COLLECTION MAPPING

OBJECT RELATION MAPPING

**Employee.java**

package com.javatpoint;

public class Employee {

private int employeeId;

private String name,email;

private Address address;

//setters and getters

}

#### Address.java

#### package com.javatpoint;

#### 

#### public class Address {

#### private int addressId;

#### private String addressLine1,city,state,country;

#### private int pincode;

#### private Employee employee;

#### //setters and getters

#### }

#### employee.hbm.xml

#### <?xml version='1.0' encoding='UTF-8'?>

#### <!DOCTYPE hibernate-mapping PUBLIC

#### "-//Hibernate/Hibernate Mapping DTD 5.3//EN"

#### "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

#### 

#### <hibernate-mapping>

#### <class name="com.javatpoint.Employee" table="emp212">

#### <id name="employeeId">

#### <generator class="increment"></generator>

#### </id>

#### <property name="name"></property>

#### <property name="email"></property>

#### 

#### <one-to-one name="address" cascade="all"></one-to-one>

#### </class>

#### 

#### </hibernate-mapping>

#### address.hbm.xml

#### <?xml version='1.0' encoding='UTF-8'?>

#### <!DOCTYPE hibernate-mapping PUBLIC

#### "-//Hibernate/Hibernate Mapping DTD 5.3//EN"

#### "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

#### 

#### <hibernate-mapping>

#### <class name="com.javatpoint.Address" table="address212">

#### <id name="addressId">

#### <generator class="foreign">

#### <param name="property">employee</param>

#### </generator>

#### </id>

#### <property name="addressLine1"></property>

#### <property name="city"></property>

#### <property name="state"></property>

#### <property name="country"></property>

#### <property name="pincode"></property>

#### 

#### <one-to-one name="employee"></one-to-one>

#### </class>

#### 

#### </hibernate-mapping>

#### hibernate.cfg.xml

#### <?xml version='1.0' encoding='UTF-8'?>

#### <!DOCTYPE hibernate-configuration PUBLIC

#### "-//Hibernate/Hibernate Configuration DTD 5.3//EN"

#### "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

#### 

#### <hibernate-configuration>

#### 

#### <session-factory>

#### <property name="hbm2ddl.auto">update</property>

#### <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

#### <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

#### <property name="connection.username">system</property>

#### <property name="connection.password">jtp</property>

#### <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

#### <mapping resource="employee.hbm.xml"/>

#### <mapping resource="address.hbm.xml"/>

#### </session-factory>

#### 

#### </hibernate-configuration>

#### Store.java

#### package com.javatpoint;

#### 

#### import org.hibernate.\*;

#### import org.hibernate.boot.Metadata;

#### import org.hibernate.boot.MetadataSources;

#### import org.hibernate.boot.registry.StandardServiceRegistry;

#### import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

#### 

#### public class Store {

#### public static void main(String[] args) {

#### 

#### StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

#### Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

#### 

#### SessionFactory factory=meta.getSessionFactoryBuilder().build();

#### Session session=factory.openSession();

#### 

#### Transaction t=session.beginTransaction();

#### 

#### Employee e1=new Employee();

#### e1.setName("Ravi Malik");

#### e1.setEmail("ravi@gmail.com");

#### 

#### Address address1=new Address();

#### address1.setAddressLine1("G-21,Lohia nagar");

#### address1.setCity("Ghaziabad");

#### address1.setState("UP");

#### address1.setCountry("India");

#### address1.setPincode(201301);

#### 

#### e1.setAddress(address1);

#### address1.setEmployee(e1);

#### 

#### session.persist(e1);

#### t.commit();

#### 

#### session.close();

#### System.out.println("success");

#### } }

### **Output**

#### Hibernate One to One Example 1 Hibernate One to One Example 2

#### Fetch.java

#### package com.javatpoint;

#### 

#### import java.util.\*;

#### import javax.persistence.TypedQuery;

#### import org.hibernate.Session;

#### import org.hibernate.SessionFactory;

#### import org.hibernate.boot.Metadata;

#### import org.hibernate.boot.MetadataSources;

#### import org.hibernate.boot.registry.StandardServiceRegistry;

#### import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

#### 

#### public class Fetch {

#### public static void main(String[] args) {

#### StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

#### Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

#### 

#### SessionFactory factory=meta.getSessionFactoryBuilder().build();

#### Session session=factory.openSession();

#### 

#### TypedQuery query=session.createQuery("from Employee e");

#### List<Employee> list=query.getResultList();

#### 

#### Iterator<Employee> itr=list.iterator();

#### while(itr.hasNext()){

#### Employee emp=itr.next();

#### System.out.println(emp.getEmployeeId()+" "+emp.getName()+" "+emp.getEmail());

#### Address address=emp.getAddress();

#### System.out.println(address.getAddressLine1()+" "+address.getCity()+" "+

#### address.getState()+" "+address.getCountry()+" "+address.getPincode());

#### }

#### 

#### session.close();

#### System.out.println("success");

#### }

#### }

### **Output**

#### Hibernate One to One Example 3

COLLECTION MAPPING

**Question**

package com.javatpoint;

import java.util.List;

public class Question {

private int id;

private String qname;

private List<String> answers;

//getters and setters

}

**question.hbm.xml**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

<hibernate-mapping>

<class name="com.javatpoint.Question" table="q100">

<id name="id">

<generator class="increment"></generator>

</id>

<property name="qname"></property>

<list name="answers" table="ans100">

<key column="qid"></key>

<index column="type"></index>

<element column="answer" type="string"></element>

</list>

</class>

</hibernate-mapping>

**XML Configuration**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

<property name="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">system</property>

<property name="connection.password">jtp</property>

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<mapping resource="question.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**StoreData**

package com.javatpoint;

import java.util.ArrayList;

import org.hibernate.\*;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreData {

public static void main(String[] args) {

StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

Transaction t=session.beginTransaction();

ArrayList<String> list1=new ArrayList<String>();

list1.add("Java is a programming language");

list1.add("Java is a platform");

ArrayList<String> list2=new ArrayList<String>();

list2.add("Servlet is an Interface");

list2.add("Servlet is an API");

Question question1=new Question();

question1.setQname("What is Java?");

question1.setAnswers(list1);

Question question2=new Question();

question2.setQname("What is Servlet?");

question2.setAnswers(list2);

session.persist(question1);

session.persist(question2);

t.commit();

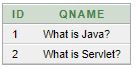
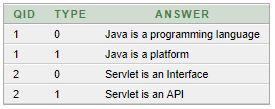
session.close();

System.out.println("success");

}

}

### **Output**

PROGRAM 8: ASSOCIATION MAPPING AND COMPONENT MAPPING

ASSOCIATION MAPPING

CREATE DATABASE AND TABLES

create database stockdb;

use stockdb;

CREATE TABLE `category` (

`category\_id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`category\_id`)

);

CREATE TABLE `product` (

`product\_id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`description` varchar(512) NOT NULL,

`price` float NOT NULL,

`category\_id` int(11) NOT NULL,

PRIMARY KEY (`product\_id`),

KEY `fk\_category` (`category\_id`),

CONSTRAINT `fk\_category` FOREIGN KEY (`category\_id`) REFERENCES `category` (`category\_id`)

);

**pom.xml**

<project

xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>net.codejava.hibernate</groupId>

<artifactId>HibernateOne2ManyAnnotationsExample</artifactId>

<version>1.0</version>

<description>Example of a Hibernate one-to-many association mapping

using annotations</description>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.2.7.SP1</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.26</version>

</dependency>

</dependencies>

</project>

**Category.java**

package net.codejava.hibernate;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

@Entity

@Table(name = "CATEGORY")

public class Category {

private long id;

private String name;

private Set<Product> products;

public Category() {

}

public Category(String name) {

this.name = name;

}

@Id

@Column(name = "CATEGORY\_ID")

@GeneratedValue

public long getId() {

return id;

}

@OneToMany(mappedBy = "category", cascade = CascadeType.ALL)

public Set<Product> getProducts() {

return products;

}

// other getters and setters...

}

**Product.java**

package net.codejava.hibernate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

@Entity

@Table(name = "PRODUCT")

public class Product {

private long id;

private String name;

private String description;

private float price;

private Category category;

public Product() {

}

public Product(String name, String description, float price,

Category category) {

this.name = name;

this.description = description;

this.price = price;

this.category = category;

}

@Id

@Column(name = "PRODUCT\_ID")

@GeneratedValue

public long getId() {

return id;

}

@ManyToOne

@JoinColumn(name = "CATEGORY\_ID")

public Category getCategory() {

return category;

}

// other getters and setters...

}

**In the Category side:**

private Set<Product> products;

@OneToMany(mappedBy = "category", cascade = CascadeType.ALL)

public Set<Product> getProducts() {

return products;

}

**In the Product side:**

private Category category;

@ManyToOne

@JoinColumn(name = "CATEGORY\_ID")

public Category getCategory() {

return category;

}

**hibernate.cfg.xml**

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/stockdb</property>

<property name="connection.username">root</property>

<property name="connection.password">secret</property>

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="show\_sql">true</property>

<mapping class="net.codejava.hibernate.Category"/>

<mapping class="net.codejava.hibernate.Product"/>

</session-factory>

</hibernate-configuration>

**StockManager.java**

package net.codejava.hibernate;

import java.util.HashSet;

import java.util.Set;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.service.ServiceRegistryBuilder;

/\*\*

\*

\* This program demonstrates using JPA annotations in Hibernate

\* in order to implement a one-to-many association mapping.

\* @author www.codejava.net

\*

\*/

public class StockManager {

public static void main(String[] args) {

// loads configuration and mappings

Configuration configuration = new Configuration().configure();

ServiceRegistryBuilder registry = new ServiceRegistryBuilder();

registry.applySettings(configuration.getProperties());

ServiceRegistry serviceRegistry = registry.buildServiceRegistry();

// builds a session factory from the service registry

SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);

// obtains the session

Session session = sessionFactory.openSession();

session.beginTransaction();

Category category = new Category("Computer");

Product pc = new Product("DELL PC", "Quad-core PC", 1200, category);

Product laptop = new Product("MacBook", "Apple High-end laptop", 2100, category);

Product phone = new Product("iPhone 5", "Apple Best-selling smartphone", 499, category);

Product tablet = new Product("iPad 3", "Apple Best-selling tablet", 1099, category);

Set<Product> products = new HashSet<Product>();

products.add(pc);

products.add(laptop);

products.add(phone);

products.add(tablet);

category.setProducts(products);

session.save(category);

session.getTransaction().commit();

session.close();

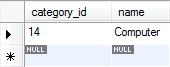
}

}

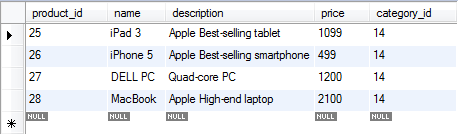
**Output**

|  |
| --- |
| Hibernate: insert into CATEGORY (name) values (?)  Hibernate: insert into PRODUCT (CATEGORY\_ID, description, name, price) values (?, ?, ?, ?)  Hibernate: insert into PRODUCT (CATEGORY\_ID, description, name, price) values (?, ?, ?, ?)  Hibernate: insert into PRODUCT (CATEGORY\_ID, description, name, price) values (?, ?, ?, ?)  Hibernate: insert into PRODUCT (CATEGORY\_ID, description, name, price) values (?, ?, ?, ?) |

**output in the category table:**



**output in the product table:**



COMPONENT MAPPING

**STUDENT**

package com.vaannila.student;

import javax.persistence.Column;

import javax.persistence.Embedded;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "STUDENT")

public class Student {

private long studentId;

private String studentName;

private Address studentAddress;

public Student() {

}

public Student(String studentName, Address studentAddress) {

this.studentName = studentName;

this.studentAddress = studentAddress;

}

@Id

@GeneratedValue

@Column(name = "STUDENT\_ID")

public long getStudentId() {

return this.studentId;

}

public void setStudentId(long studentId) {

this.studentId = studentId;

}

@Column(name = "STUDENT\_NAME", nullable = false, length = 100)

public String getStudentName() {

return this.studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

@Embedded

public Address getStudentAddress() {

return this.studentAddress;

}

public void setStudentAddress(Address studentAddress) {

this.studentAddress = studentAddress;

}

}

**Address**

package com.vaannila.student;

import javax.persistence.Column;

import javax.persistence.Embeddable;

@Embeddable

public class Address {

private String street;

private String city;

private String state;

private String zipcode;

public Address() {

}

public Address(String street, String city, String state, String zipcode) {

this.street = street;

this.city = city;

this.state = state;

this.zipcode = zipcode;

}

@Column(name = "ADDRESS\_STREET", nullable = false, length=250)

public String getStreet() {

return this.street;

}

public void setStreet(String street) {

this.street = street;

}

@Column(name = "ADDRESS\_CITY", nullable = false, length=50)

public String getCity() {

return this.city;

}

public void setCity(String city) {

this.city = city;

}

@Column(name = "ADDRESS\_STATE", nullable = false, length=50)

public String getState() {

return this.state;

}

public void setState(String state) {

this.state = state;

}

@Column(name = "ADDRESS\_ZIPCODE", nullable = false, length=10)

public String getZipcode() {

return this.zipcode;

}

public void setZipcode(String zipcode) {

this.zipcode = zipcode;

}

}

**hibernate configuration**

 <?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class"> org.hsqldb.jdbcDriver</property>

<property name="hibernate.connection.url"> jdbc:hsqldb:hsql://localhost</property>

<property name="hibernate.connection.username">sa</property>

<property name="connection.password"></property>

<property name="connection.pool\_size">1</property>

<property name="hibernate.dialect"> org.hibernate.dialect.HSQLDialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">create-drop</property>

<mapping class="com.vaannila.student.Student" />

<mapping class="com.vaannila.student.Address" />

</session-factory>

</hibernate-configuration>

**Main class**

package com.vaannila.student;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.Transaction;

import com.vaannila.util.HibernateUtil;

public class Main {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction transaction = null;

try {

transaction = session.beginTransaction();

Address address = new Address("OMR Road", "Chennai", "TN", "600097");

Student student = new Student("Eswar", address);

session.save(student);

transaction.commit();

} catch (HibernateException e) {

transaction.rollback();

e.printStackTrace();

} finally {

session.close();

}

}

}

**Output**

http://cdn.dzone.com/static/images/vaannila/hibernate/hibernateComponentPic2.gif

PROGRAM 9 :INHERITANCE MAPPING

**hibernate configuration**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

<hibernate-mapping>

<class name="com.javatpoint.mypackage.Employee" table="emp121" discriminator-value="emp">

<id name="id">

<generator class="increment"></generator>

</id>

<discriminator column="type" type="string"></discriminator>

<property name="name"></property>

<subclass name="com.javatpoint.mypackage.Regular\_Employee" discriminator-value="reg\_emp">

<property name="salary"></property>

<property name="bonus"></property>

</subclass>

<subclass name="com.javatpoint.mypackage.Contract\_Employee" discriminator-value="con\_emp">

<property name="pay\_per\_hour"></property>

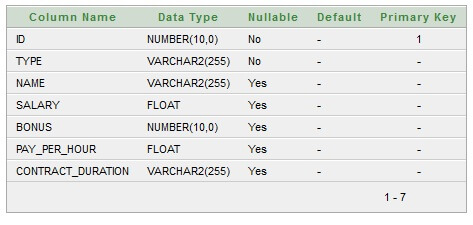
<property name="contract\_duration"></property>

</subclass>

</class>

</hibernate-mapping>

**The table structure for this hierarchy is as shown below:**



**Employee.java**

package com.javatpoint.mypackage;

public class Employee {

private int id;

private String name;

//getters and setters

}

**Regular\_Employee.java**

package com.javatpoint.mypackage;

public class Regular\_Employee extends Employee{

private float salary;

private int bonus;

//getters and setters

}

**Contract\_Employee.java**

package com.javatpoint.mypackage;

public class Contract\_Employee extends Employee{

private float pay\_per\_hour;

private String contract\_duration;

//getters and setters

}

**employee.hbm.xml**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

<hibernate-mapping>

<class name="com.javatpoint.mypackage.Employee" table="emp121" discriminator-value="emp">

<id name="id">

<generator class="increment"></generator>

</id>

<discriminator column="type" type="string"></discriminator>

<property name="name"></property>

<subclass name="com.javatpoint.mypackage.Regular\_Employee" discriminator-value="reg\_emp">

<property name="salary"></property>

<property name="bonus"></property>

</subclass>

<subclass name="com.javatpoint.mypackage.Contract\_Employee" discriminator-value="con\_emp">

<property name="pay\_per\_hour"></property>

<property name="contract\_duration"></property>

</subclass>

</class>

</hibernate-mapping>

### **Add mapping of hbm file in configuration file**

<mapping resource="employee.hbm.xml"/>

**hibernate.cfg.xml**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

<property name="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">system</property>

<property name="connection.password">jtp</property>

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<mapping resource="employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**StoreData.java**

package com.javatpoint.mypackage;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreData {

public static void main(String[] args) {

StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

Transaction t=session.beginTransaction();

Employee e1=new Employee();

e1.setName("Gaurav Chawla");

Regular\_Employee e2=new Regular\_Employee();

e2.setName("Vivek Kumar");

e2.setSalary(50000);

e2.setBonus(5);

Contract\_Employee e3=new Contract\_Employee();

e3.setName("Arjun Kumar");

e3.setPay\_per\_hour(1000);

e3.setContract\_duration("15 hours");

session.persist(e1);

session.persist(e2);

session.persist(e3);

t.commit();

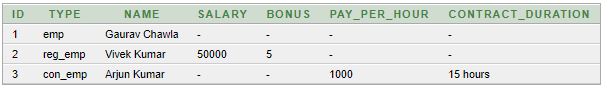
session.close();

System.out.println("success");

}

}

### **Output:**



PROGRAM 10: SPRING ACTION AND SPRING MVC

**Student.java**

package com.tutorialspoint;

public class Student {

private Integer age;

private String name;

private Integer id;

public void setAge(Integer age) {

this.age = age;

}

public Integer getAge() {

return age;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setId(Integer id) {

this.id = id;

}

public Integer getId() {

return id;

}

}

**StudentController.java**

package com.tutorialspoint;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.ui.ModelMap;

@Controller

public class StudentController {

@RequestMapping(value = "/student", method = RequestMethod.GET)

public ModelAndView student() {

return new ModelAndView("student", "command", new Student());

}

@RequestMapping(value = "/addStudent", method = RequestMethod.POST)

public String addStudent(@ModelAttribute("SpringWeb")Student student,

ModelMap model) {

model.addAttribute("name", student.getName());

model.addAttribute("age", student.getAge());

model.addAttribute("id", student.getId());

return "result";

}

}

**web.xml**

<web-app id = "WebApp\_ID" version = "2.4"

xmlns = "http://java.sun.com/xml/ns/j2ee"

xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation = "http://java.sun.com/xml/ns/j2ee

http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">

<display-name>Spring MVC Form Handling</display-name>

<servlet>

<servlet-name>HelloWeb</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>HelloWeb</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

**HelloWeb-servlet.xml**

<beans xmlns = "http://www.springframework.org/schema/beans"

xmlns:context = "http://www.springframework.org/schema/context"

xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation = "http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:component-scan base-package = "com.tutorialspoint" />

<bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name = "prefix" value = "/WEB-INF/jsp/" />

<property name = "suffix" value = ".jsp" />

</bean>

</beans>

**student.jsp**

<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>

<html>

<head>

<title>Spring MVC Form Handling</title>

</head>

<body>

<h2>Student Information</h2>

<form:form method = "POST" action = "/HelloWeb/addStudent">

<table>

<tr>

<td><form:label path = "name">Name</form:label></td>

<td><form:input path = "name" /></td>

</tr>

<tr>

<td><form:label path = "age">Age</form:label></td>

<td><form:input path = "age" /></td>

</tr>

<tr>

<td><form:label path = "id">id</form:label></td>

<td><form:input path = "id" /></td>

</tr>

<tr>

<td colspan = "2">

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

</td>

</tr>

</table>

</form:form>

</body>

</html>

**result.jsp**

<%@page contentType = "text/html;charset = UTF-8" language = "java" %>

<%@page isELIgnored = "false" %>

<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>

<html>

<head>

<title>Spring MVC Form Handling</title>

</head>

<body>

<h2>Submitted Student Information</h2>

<table>

<tr>

<td>Name</td>

<td>${name}</td>

</tr>

<tr>

<td>Age</td>

<td>${age}</td>

</tr>

<tr>

<td>ID</td>

<td>${id}</td>

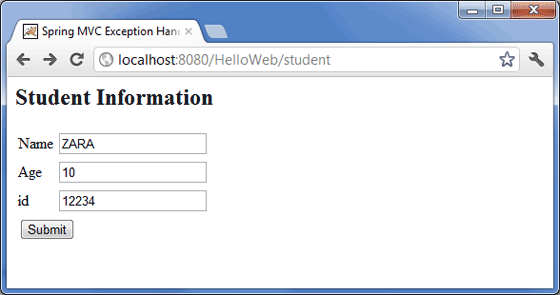
</tr>

</table>

</body>

</html>

**OUTPUT 1:**



**OUTPUT 2:**

