**EXPERIMENT-8**

**AIM:Write a chat application using web sockets.**

**Source code:**

Index.html

<!DOCTYPE html>

<html>

<head>

<title></title>

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

<script src="websocket.js"></script>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div id="wrapper">

<h1>Java Websocket Home</h1>

<p>Welcome to the Java WebSocket Home. Click the Add a device button to start adding devices.</p>

<br />

<div id="addDevice">

<div class="button"> <a href="#" OnClick="showForm()">Add a device</a> </div>

<form id="addDeviceForm">

<h3>Add a new device</h3>

<span>Name: <input type="text" name="device\_name" id="device\_name"></span>

<span>Type:

<select id="device\_type">

<option name="type" value="Appliance">Appliance</option>

<option name="type" value="Electronics">Electronics</option>

<option name="type" value="Lights">Lights</option> 83

<option name="type" value="Other">Other</option>

</select></span>

<span>Description:<br />

<textarea name="description" id="device\_description" rows="2" cols="50"></textarea>

</span>

<input type="button" class="button" value="Add" onclick=formSubmit()>

<input type="reset" class="button" value="Cancel" onclick=hideForm()>

</form>

</div>

<br />

<h3>Currently connected devices:</h3>

<div id="content">

</div>

</div>

</body>

</html>

**Device.java**

package org.example.model;

public class Device {

private int id;

private String name;

private String status;

private String type;

private String description;

public Device() {

}

public int getId() {

return id; 84

}

public String getName() {

return name;

}

public String getStatus() {

return status;

}

public String getType() {

return type;

}

public String getDescription() {

return description;

}

public void setId(int id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public void setStatus(String status) {

this.status = status;

}

public void setType(String type) {

this.type = type;

}

public void setDescription(String description) {

this.description = description;

}

**DeviceWebSocketServer.java**

package org.example.websocket;

import javax.websocket.OnClose; 85

import javax.websocket.OnError;

import javax.websocket.OnMessage;

import javax.websocket.OnOpen;

import javax.websocket.Session;

import javax.websocket.server.ServerEndpoint;

import javax.enterprise.context.ApplicationScoped;

import javax.inject.Inject;

import java.io.StringReader;

import javax.json.Json;

import javax.json.JsonObject;

import javax.json.JsonReader;

import org.example.model.Device;

import java.util.logging.Level;

import java.util.logging.Logger;

@ApplicationScoped

@ServerEndpoint("/actions")

public class DeviceWebSocketServer {

@Inject

private DeviceSessionHandler sessionHandler;

@OnOpen

public void open(Session session) {

sessionHandler.addSession(session);

}

@OnClose

public void close(Session session) {

sessionHandler.removeSession(session);

}

@OnError

public void onError(Throwable error) {

Logger.getLogger(DeviceWebSocketServer.class.getName()).log(Level.SEVERE, null, error); 86

}

@OnMessage

public void handleMessage(String message, Session session) {

try (JsonReader reader = Json.createReader(new StringReader(message))) {

JsonObject jsonMessage = reader.readObject();

if ("add".equals(jsonMessage.getString("action"))) {

Device device = new Device();

device.setName(jsonMessage.getString("name"));

device.setDescription(jsonMessage.getString("description"));

device.setType(jsonMessage.getString("type"));

device.setStatus("Off");

sessionHandler.addDevice(device);

}

if ("remove".equals(jsonMessage.getString("action"))) {

int id = (int) jsonMessage.getInt("id");

sessionHandler.removeDevice(id);

}

if ("toggle".equals(jsonMessage.getString("action"))) {

int id = (int) jsonMessage.getInt("id");

sessionHandler.toggleDevice(id);

}

}

}

}

**DeviceSessionHandler.java**

package org.example.websocket;

import java.io.IOException;

import java.util.ArrayList;

import javax.enterprise.context.ApplicationScoped;

import java.util.HashSet;

import java.util.List; 87

import java.util.Set;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.json.JsonObject;

import javax.json.spi.JsonProvider;

import javax.websocket.Session;

import org.example.model.Device;

@ApplicationScoped

public class DeviceSessionHandler {

private int deviceId = 0;

private final Set<Session> sessions = new HashSet<>();

private final Set<Device> devices = new HashSet<>();

public void addSession(Session session) {

sessions.add(session);

}

public void removeSession(Session session) {

sessions.remove(session);

for (Device device : devices) {

JsonObject addMessage = createAddMessage(device);

sendToSession(session, addMessage);

}

}

public List<Device> getDevices() {

return new ArrayList<>(devices);

}

public void addDevice(Device device) {

device.setId(deviceId);

devices.add(device); 88

deviceId++;

JsonObject addMessage = createAddMessage(device);

sendToAllConnectedSessions(addMessage);

}

public void removeDevice(int id) {

Device device = getDeviceById(id);

if (device != null) {

devices.remove(device);

JsonProvider provider = JsonProvider.provider();

JsonObject removeMessage = provider.createObjectBuilder()

.add("action", "remove")

.add("id", id)

.build();

sendToAllConnectedSessions(removeMessage);

}

}

public void toggleDevice(int id) {

JsonProvider provider = JsonProvider.provider();

Device device = getDeviceById(id);

if (device != null) {

if ("On".equals(device.getStatus())) {

device.setStatus("Off");

} else {

device.setStatus("On");

}

JsonObject updateDevMessage = provider.createObjectBuilder()

.add("action", "toggle")

.add("id", device.getId())

.add("status", device.getStatus())

.build();

sendToAllConnectedSessions(updateDevMessage); 89

}

}

private Device getDeviceById(int id) {

for (Device device : devices) {

if (device.getId() == id) {

return device;

}

}

return null;

}

private JsonObject createAddMessage(Device device) {

JsonProvider provider = JsonProvider.provider();

JsonObject addMessage = provider.createObjectBuilder()

.add("action", "add")

.add("id", device.getId())

.add("name", device.getName())

.add("type", device.getType())

.add("status", device.getStatus())

.add("description", device.getDescription())

.build();

return addMessage;

}

private void sendToAllConnectedSessions(JsonObject message) {

for (Session session : sessions) {

sendToSession(session, message);

}

}

private void sendToSession(Session session, JsonObject message) {

try { 90

session.getBasicRemote().sendText(message.toString());

} catch (IOException ex) {

sessions.remove(session);

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

}

}

}

**style.css**

body {

font-family: Arial, Helvetica, sans-serif;

font-size: 80%;

background-color: #1f1f1f;

}

#wrapper {

width: 960px;

margin: auto;

text-align: left;

color: #d9d9d9;

}

p {

text-align: left;

}

.button {

display: inline;

color: #fff;

background-color: #f2791d;

padding: 8px;

margin: auto;

border-radius: 8px; 91

-moz-border-radius: 8px;

-webkit-border-radius: 8px;

box-shadow: none;

border: none;

}

.button:hover {

background-color: #ffb15e;

}

.button a, a:visited, a:hover, a:active {

color: #fff;

text-decoration: none;

}

#addDevice {

text-align: center;

width: 960px;

margin: auto;

margin-bottom: 10px;

}

#addDeviceForm {

text-align: left;

width: 400px;

margin: auto;

padding: 10px;

}

#addDeviceForm span {

display: block;

}

#content {

margin: auto;

width: 960px; 92

}

.device {

width: 180px;

height: 110px;

margin: 10px;

padding: 16px;

color: #fff;

vertical-align: top;

border-radius: 8px;

-moz-border-radius: 8px;

-webkit-border-radius: 8px;

display: inline-block;

}

.device.off {

background-color: #c8cccf;

}

.device span {

display: block;

}

.deviceName {

text-align: center;

font-weight: bold;

margin-bottom: 12px;

}

.removeDevice {

margin-top: 12px;

text-align: center;

}

.device.Appliance { 93

background-color: #5eb85e;

}

.device.Appliance a:hover {

color: #a1ed82;

}

.device.Electronics {

background-color: #0f90d1;

}

.device.Electronics a:hover {

color: #4badd1;

}

.device.Lights {

background-color: #c2a00c;

}

.device.Lights a:hover {

color: #fad232;

}

.device.Other {

background-color: #db524d;

}

.device.Other a:hover {

color: #ff907d;

}

.device a {

text-decoration: none;

}

.device a:visited, a:active, a:hover {

color: #fff;

} 94

.device a:hover {

text-decoration: underline;

**websocket.js**

window.onload = init;

var socket = new WebSocket("ws://localhost:8080/WebsocketHome/actions");

socket.onmessage = onMessage;

function onMessage(event) {

var device = JSON.parse(event.data);

if (device.action === "add") {

printDeviceElement(device);

}

if (device.action === "remove") {

document.getElementById(device.id).remove();

//device.parentNode.removeChild(device);

}

if (device.action === "toggle") {

var node = document.getElementById(device.id);

var statusText = node.children[2];

if (device.status === "On") {

statusText.innerHTML = "Status: " + device.status + " (<a href=\"#\" OnClick=toggleDevice(" + device.id + ")>Turn off</a>)";

} else if (device.status === "Off") {

statusText.innerHTML = "Status: " + device.status + " (<a href=\"#\" OnClick=toggleDevice(" + device.id + ")>Turn on</a>)";

}

}

}

function addDevice(name, type, description) {

var DeviceAction = {

action: "add", 95

name: name,

type: type,

description: description

};

socket.send(JSON.stringify(DeviceAction));

}

function removeDevice(element) {

var id = element;

var DeviceAction = {

action: "remove",

id: id

};

socket.send(JSON.stringify(DeviceAction));

}

function toggleDevice(element) {

var id = element;

var DeviceAction = {

action: "toggle",

id: id

};

socket.send(JSON.stringify(DeviceAction));

}

function printDeviceElement(device) {

var content = document.getElementById("content");

var deviceDiv = document.createElement("div");

deviceDiv.setAttribute("id", device.id);

deviceDiv.setAttribute("class", "device " + device.type);

content.appendChild(deviceDiv);

var deviceName = document.createElement("span");

deviceName.setAttribute("class", "deviceName"); 96

deviceName.innerHTML = device.name;

deviceDiv.appendChild(deviceName);

var deviceType = document.createElement("span");

deviceType.innerHTML = "<b>Type:</b> " + device.type;

deviceDiv.appendChild(deviceType);

var deviceStatus = document.createElement("span");

if (device.status === "On") {

deviceStatus.innerHTML = "<b>Status:</b> " + device.status + " (<a href=\"#\" OnClick=toggleDevice(" + device.id + ")>Turn off</a>)";

} else if (device.status === "Off") {

deviceStatus.innerHTML = "<b>Status:</b> " + device.status + " (<a href=\"#\" OnClick=toggleDevice(" + device.id + ")>Turn on</a>)";

//deviceDiv.setAttribute("class", "device off");

}

deviceDiv.appendChild(deviceStatus);

var deviceDescription = document.createElement("span");

deviceDescription.innerHTML = "<b>Comments:</b> " + device.description;

deviceDiv.appendChild(deviceDescription);

var removeDevice = document.createElement("span");

removeDevice.setAttribute("class", "removeDevice");

removeDevice.innerHTML = "<a href=\"#\" OnClick=removeDevice(" + device.id + ")>Remove device</a>";

deviceDiv.appendChild(removeDevice);

}

function showForm() {

document.getElementById("addDeviceForm").style.display = '';

}

function hideForm() {

document.getElementById("addDeviceForm").style.display = "none";

}

function formSubmit() {

var form = document.getElementById("addDeviceForm"); 97

var name = form.elements["device\_name"].value;

var type = form.elements["device\_type"].value;

var description = form.elements["device\_description"].value;

hideForm();

document.getElementById("addDeviceForm").reset();

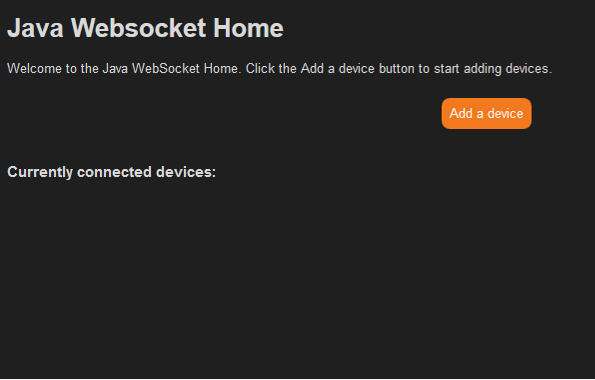
addDevice(name, type, description);

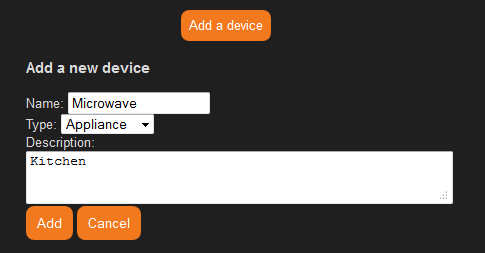
}

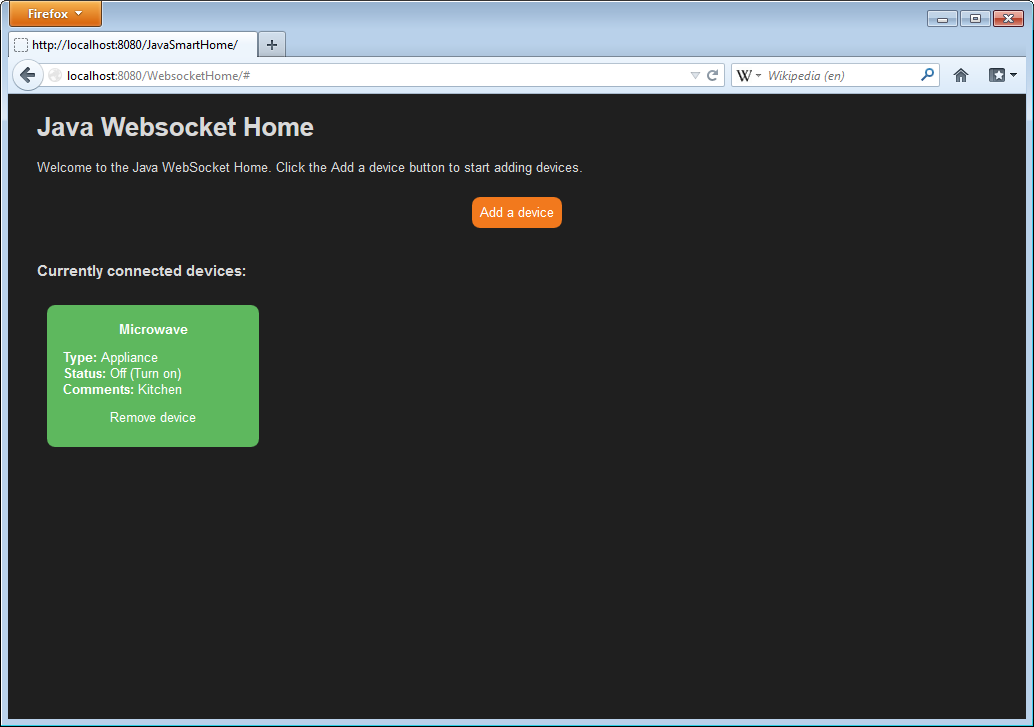
function init() {

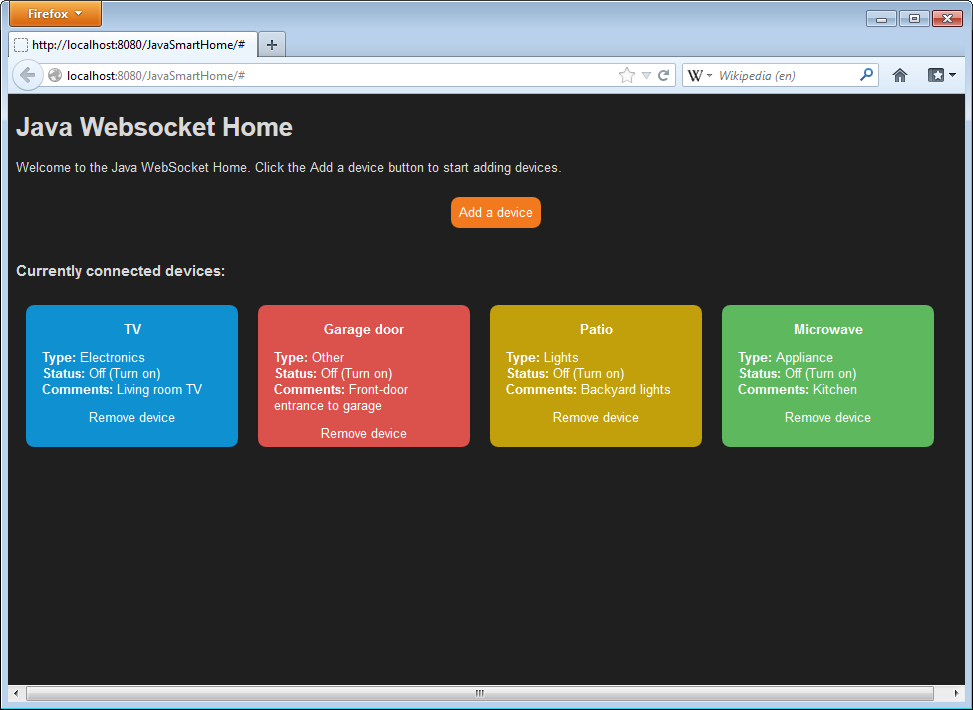
hideForm();

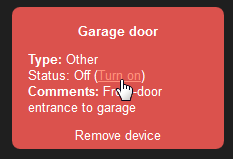
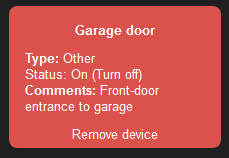
}



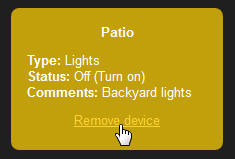
 98



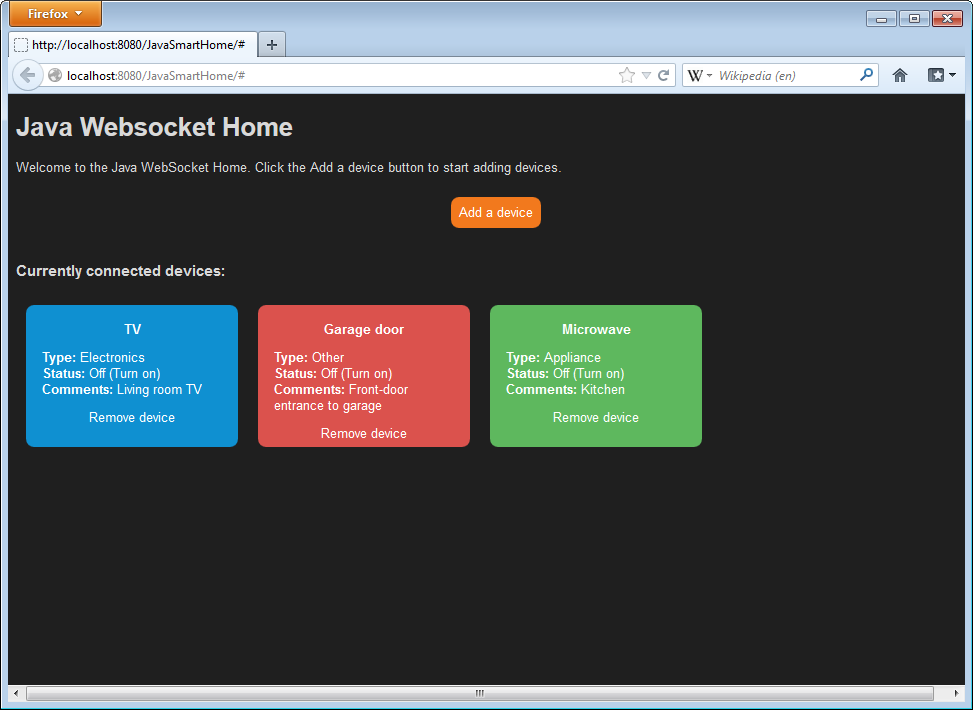




99



100



101

**EXPERIMENT-9**

**AIM:Write an application to demonstrate Session Bean Entity Bean(Persistence).**

**Source code:**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Add Employee</title>

</head>

<body>

<h2>Add Employee</h2>

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

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

Position: <input type="text" name="position" required /><br>

Salary: <input type="text" name="salary" required /><br>

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

</form>

</body>

</html>

**EmployeeServlet.java**

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;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List; 102

@WebServlet("/EmployeeServlet")

public class EmployeeServlet extends HttpServlet {

@EJB

private EmployeeSessionBean employeeSessionBean;

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

List<Employee> employees = employeeSessionBean.getAllEmployees();

response.setContentType("text/html");

PrintWriter out = response.getWriter();

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

out.println("<h2>Employee List</h2>");

for (Employee employee : employees) {

out.println("<p>" + employee.getName() + " - " + employee.getPosition() + " - " + employee.getSalary() + "</p>");

}

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

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

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

String position = request.getParameter("position");

double salary = Double.parseDouble(request.getParameter("salary"));

employeeSessionBean.addEmployee(name, position, salary);

response.sendRedirect("EmployeeServlet");

}

}

**Employee.java**

import java.io.Serializable;

public class Employee implements Serializable { 103

private Long id;

private String name;

private String position;

private double salary;

// Constructor

public Employee(Long id, String name, String position, double salary) {

this.id = id;

this.name = name;

this.position = position;

this.salary = salary;

}

// Getters and Setters

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 getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public double getSalary() { 104

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

**EmployeeSessionBean.java**

import javax.ejb.Stateless;

import java.util.ArrayList;

import java.util.List;

@Stateless

public class EmployeeSessionBean {

// In-memory list to mimic a database

private List<Employee> employeeList = new ArrayList<>();

private static Long idCounter = 1L;

// Method to add employee

public void addEmployee(String name, String position, double salary) {

Employee employee = new Employee(idCounter++, name, position, salary);

employeeList.add(employee);

}

// Method to get all employees

public List<Employee> getAllEmployees() {

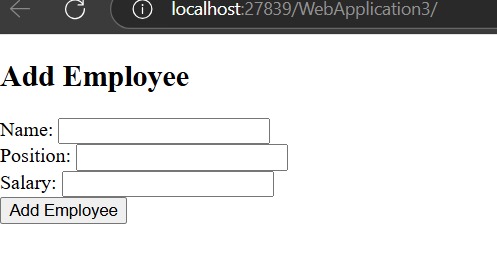
return employeeList;

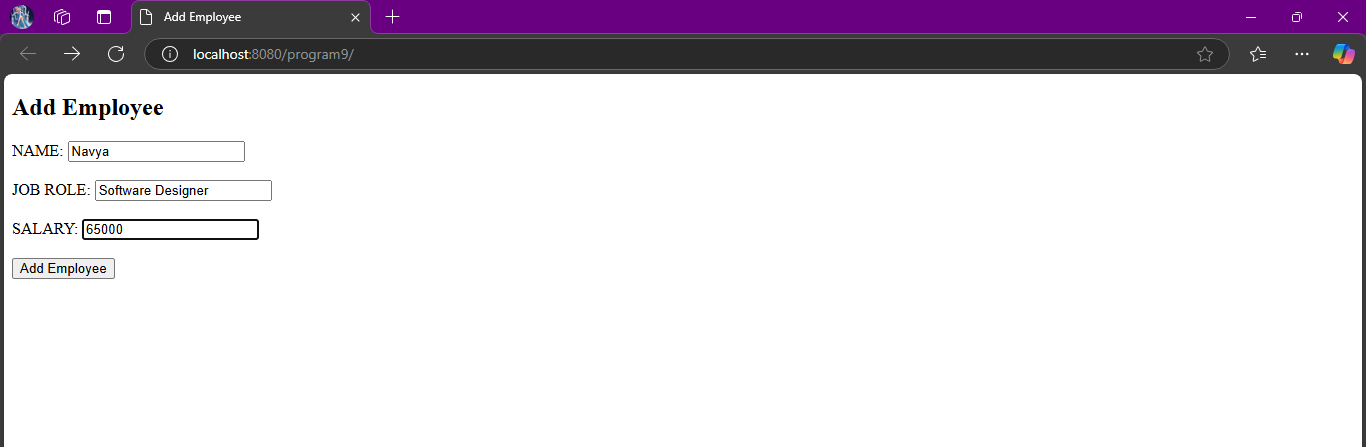
}

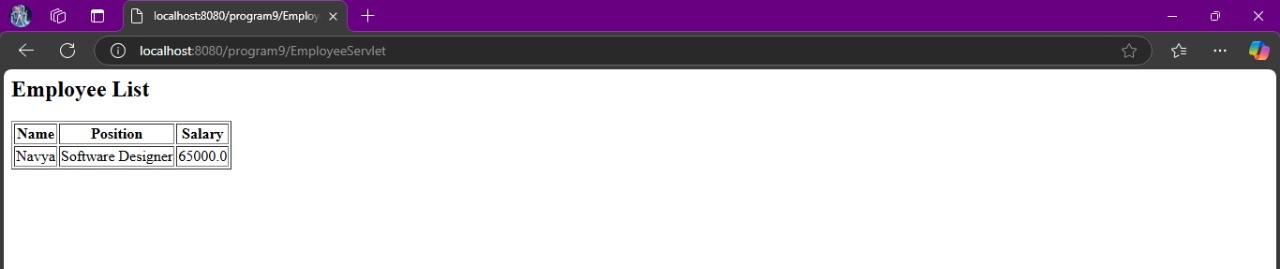
}

105

**Output:**



****



106